Lassi Paavolainen

Kongressinhallintajärjestelmän mallintaminen ja analysointi väritetyillä Petri-verkoilla

Tietotekniikan (ohjelmistotekniikka)
pro gradu -tutkielma
25. toukokuuta 2007

Jyväskylän yliopisto

Tietotekniikan laitos

Jyväskylä

Tekijä: Lassi Paavolainen
Yhteystiedot: lopaavol@jyu.fi
Työn nimi: Kongressinhallintajärjestelmän mallintaminen ja analysointi väritetyillä Petri-verkoilla
Title in English: Modelling and Analysing of the Congress Management System using Coloured Petri Nets
Työ: Tietotekniikan (ohjelmistotekniikka) pro gradu -tutkielma
Sivumäärä: 138
Tiivistelmä: Kongressinhallintajärjestelmä on Jyväskylän yliopiston tietotekniikan laitoksella kehitetty järjestelmä tieteellisen kongressin hallintaan. Järjestelmää on kehitetty esiin tulleiden tarpeiden mukaan ilman pitkäjänteisempää suunnittelua. Tästä syystä tutkimuksessa mallinnetaan ja analysoidaan järjestelmän tärkeintä kokonaisuutta, abstraktien vastaanotto ja arviointi -prosessia.

Abstraktien vastaanotto ja arviointi -prosessia mallinnetaan ja analysoidaan väritetyillä Petri-verkoilla. Tutkimuksessa kuvataan sekä väritettyjen että perinteisten Petri-verkkojen määrittelyt, ja esitellään niiden käyttöä järjestelmien analysoinnissa. Abstraktien vastaanotto ja arviointi -prosessin mallia analysoidaan simuloimalla ja tila-avaruuden avulla. Mallin analysoinnin aikana havaittiin muutama suunnittelematon ja virheellinen toiminnallisuus, mutta pääasiallisesti prosessin malli todettiin toimivaksi.

English abstract: The Congress Management System is a system developed in the Department of Mathematical Information Technology at University of Jyväskylä for managing a scientific congress. The system has been developed at need without a long term planning. This is the reason why the system's most important component, the process of receiving and reviewing of the abstracts, is modelled and analysed in this research.

The process of receiving and reviewing of the abstracts is modelled and analysed using coloured Petri nets. The research contains definitions of both coloured and traditional Petri nets, and it introduces analysis of systems using Petri nets. The model of the process of receiving and reviewing of the abstracts is analysed by simulation and using the state space of the model. Few unplanned and faulty functionalities were found during the analysis of the model, but mainly the model of the process was found functional.

Avainsanat: Petri-verkot, Väritetyt Petri-verkot, Kongressinhallintajärjestelmä, Hierarkkiset väritetyt Petri-verkot, Järjestelmien analysointi, Järjestelmän tila-avaruus, Tapahtumagraafi, Invariantit.
Keywords: Petri nets, Coloured Petri nets, The Congress Management System, Hierarchical coloured Petri nets, System analysis, State space of system, Occurence Graph, Invariants.

Esipuhe

Seitsemän kuukauden pro gradu -projekti on nyt valmis ja kansissa. Alkuvuosi 2007 kului mielenkiintoisissa merkeissä töitä ja pro gradua tehdessä. Alunperin hieman tiiviimmäksi suunnittelemani pro gradu pääsi kevään aikana laajenemaan suunnitellusta, ja näin myös pro gradun valmiiksi saaminen venyi. Jälkeenpäin ajateltuna pro gradun laajeneminen oli vain hyvä asia, sillä näin pääsin perehtymään entistä paremmin Petri-verkkojen ja järjestelmien analysoinnin mielenkiintoiseen maailmaan. Kevät oli välillä raskas, mutta nyt voi olla tyytyväinen työn tulokseen. Parannettavaa toki löytyy aina. Pro gradusta olenkin saanut tärkeää kokemusta mahdollisia tulevia tutkimuksia varten.

Haluaisin kiittää Jyväskylän yliopiston tietotekniikan laitosta mielenkiintoisista töistä Kongressinhallintajärjestelmän parissa. Erityisesti haluaisin kiittää Tuomo Rossia, joka on järjestelmän kehityksessä ollut alusta asti mukana. Tuomolle menee kiitos myös ideasta käyttää Petri-verkkoja Kongressinhallintajärjestelmän mallintamisessa ja analysoinnissa. Lisäksi Tuomo toimi pro graduni toisena ohjaajana. Antti-Juhani Kaijanaho ansaitsee myös kiitokseni pro graduni ohjauksesta. Kiitokset menevät myös kaikille niille, jotka ovat tutkielmaani lukeneet ja kommentoineet.

Lämpimät kiitokset Anitalle pro graduni aikana saamastani tuesta ja mielenkiinnosta tutkielman tekoa kohtaan. Ilman Anitaa kevät olisi ollut entistä raskaampi. Lopuksi vielä kiitokset vanhemmilleni, sillä ilman heitä ei olisi minuakaan.

Jyväskylässä 25.5.2007

Lassi Paavolainen

Sisältö

1 Johdanto

Tutkimuksen tarkoituksena on mallintaa Kongressinhallintajärjestelmän abstraktien vastaanotto ja arviointi -prosessia, ja analysoida prosessin toiminnallisuutta mallin avulla. Kongressinhallintajärjestelmä on Jyväskylän yliopiston tietotekniikan laitoksella kehitetty järjestelmä tieteellisen kongressin hallintaan. Järjestelmää on kehitetty vuodesta 2004 lähtien, ja alkuvuoden 2007 aikana sen avulla on hallittu kahta Jyväskylän yliopistolla järjestettävää tieteellistä kongressia.

Kongressinhallintajärjestelmän kehitys on tapahtunut kolmessa tietotekniikan laitoksen opiskelijaprojektissa sekä erillisissä työjaksoissa. Näiden aikana järjestelmään on kehitetty uusia kokonaisuuksia tarpeiden mukaan. Järjestelmästä ei kuitenkaan ole ollut yhtä selkeää tulevan kehityksen suunnitelmaa. Tästä johtuen uusien ominaisuuksien vaikutus aikaisemmin kehitettyihin on ollut tiedossa vain kehittäjien ajatuksissa.

Kongressinhallintajärjestelmän toiminnasta ei ole olemassa nykyisiä ominaisuuksia vastaavia malleja. Tämän takia järjestelmän analysointi on tapahtunut lähinnä lähdekoodia tutkimalla. Lähdekoodin tutkimisella on kuitenkin hankalaa saada selkeää kuvaa kokonaisuuksista, jotka erilaisilla mallinnusmenetelmillä voidaan helposti mallittaa.

Mallinnusmenetelmiä on kehitetty lukuisia erilaisia. Yksikään menetelmä ei kuitenkaan ole jokaisessa käyttötarkoituksessa muita parempi. Monesti parhaaseen lopputulokseen päästään käyttämällä useampaa menetelmää yhtäaikaisesti. Abstraktien vastaanotto ja arviointi -prosessi on luonteeltaan dynaaminen. Sen keskeisessä osassa ovat abstraktien tilasiirtymät järjestelmän sisällä. Tästä syystä mallinnusmenetelmän vahvuuksiin tulisi kuulua dynaamisten ominaisuuksien mallintaminen.

Petri-verkot on formaali mallinnusmenetelmä diskreettien hajautettujen järjestelmien mallintamiseen ja analysointiin. Petri-verkot tarjoavat sekä formaalisti määritellyn matemaattisen perustan että havainnollisen graafisen esitystavan. Formaalien perusteiden ansiosta Petri-verkkojen analysointiin on helppoa kehittää työkaluja. Lisäksi graafinen esitystapa antaa järjestelmän mallintajalle selkeän näkymän verkon rakentamiseen niin, ettei kaikkia Petri-verkkojen matemaattisia ominaisuuksia tarvitsisi edes tuntea. Näin Petri-verkkojen formaaleja ominaisuuksia päästään

käyttämään jo suhteellisen lyhyellä perehdytyksellä.

Väritetyt Petri-verkot ovat yksi Petri-verkkojen alaluokka. Väritetyt Petri-verkot kuuluvat niin sanottuihin korkean luokan Petri-verkkoihin, sillä ne sisältävät mahdollisuuden käyttää mallintajan määriteltävissä olevia tyyppejä. Tämän ansiosta väritetyt Petri-verkot ovat ilmaisuvoimaltaan perinteisiä Petri-verkkoja parempia.

Petri-verkkojen vahvuutena on niiden dynaamisuus. Mallinnettua Petri-verkkoa voidaan suorittaa kuten ohjelmaa. Tämä antaa järjestelmän suunnittelijalle mahdollisuuden simuloida suunnitelmaansa, ja havaita suorituksen aikana ilmenneitä ongelmia. Mallin pohjalta voidaan myös luoda järjestelmän tila-avaruus, jota voidaan kehitettyjen työkalujen avulla analysoida automaattisesti.

Petri-verkkojen dynaamisuuden ja väritettyjen Petri-verkkojen tarjoaman tyyppijärjestelmän ansiosta ne soveltuvat hyvin Kongressinhallintajärjestelmän abstraktien vastaanotto ja arviointi -prosessin mallintamiseen. Tutkimuksessa mallinnetaan väritetyillä Petri-verkoilla abstraktien vastaanotto ja arviointi -prosessia, ja analysoidaan luotuja malleja sekä simuloimalla että tila-avaruuden avulla. Tutkimuksen tarkoituksena on löytää prosessissa mahdollisesti olevia virheitä, ja tarjota kehittäjille ideoita parannusta vaativista kohdista. Lisäksi tarkoituksena on pyrkiä varmistamaan prosessin mahdollisimman oikeellinen toiminta.

Tutkielma etenee seuraavassa järjestyksessä. Luvussa 2 esitellään Kongressinhallintajärjestelmän rakenne, kehityshistoria, ja abstraktien vastaanotto ja arviointi -prosessi. Näiden avulla esitetään myös Kongressinhallintajärjestelmään liittyvä tutkimusongelma. Luvussa 3 esitetään Petri-verkkojen periaatteet sekä niiden käyttökohteet. Lisäksi määritellään Petri-verkoilla analysoitavia järjestelmien ominaisuuksia, ja analysointia varten kehitettyjä menetelmiä.

Luvussa 4 esitellään tutkimuksessa käytetyt väritetyt Petri-verkot sekä niiden avulla määriteltävät hierarkkiset väritetyt Petri-verkot. Lisäksi luvussa 4 annetaan esimerkkejä väritettyjen Petri-verkkojen käytöstä osana ohjelmistokehitystä sekä esitetään väritetyille Petri-verkoille kehitettyjä analysointimenetelmiä.

Tutkielman kokeellisessa osassa mallinnetaan ja analysoidaan Kongressinhallintajärjestelmän abstraktien vastaanotto ja arviointi -prosessia. Luvussa 5 on esitetty mallinnukseen käytetyt ohjelmistot ja laitteistot. Tämän lisäksi luvussa käydään läpi mallinnetut väritetyt Petri-verkot. Luvussa 6 esitetään mallinnettujen verkkojen pohjalta sekä simuloinnin että tila-avaruuden avulla tehtyjä analysointituloksia.

2 Kongressinhallintajärjestelmän esittely

Kongressinhallintajärjestelmä on Jyväskylän yliopiston tietotekniikan laitoksella kehitetty järjestelmä tieteellisen kongressin hallintaan. Järjestelmällä voidaan vastaanottaa ja arvioida kongressin abstraktit, sekä luoda kongressin aikataulutus. Näiden toimintojen lisäksi abstraktien, kokonaisten artikkeleiden ja aikataulutuksen pohjalta voidaan generoida kongressin kokoomajulkaisu ja abstraktikirja.

Luvussa esitellään Kongressinhallintajärjestelmän yleinen rakenne sekä kehityshistoria. Näiden lisäksi luvussa annetaan tarkempi kuvaus järjestelmän abstraktien vastaanotto ja arviointi -prosessista, sekä määritellään tähän liittyvä tutkimusongelma.

2.1 Kongressinhallintajärjestelmän rakenne

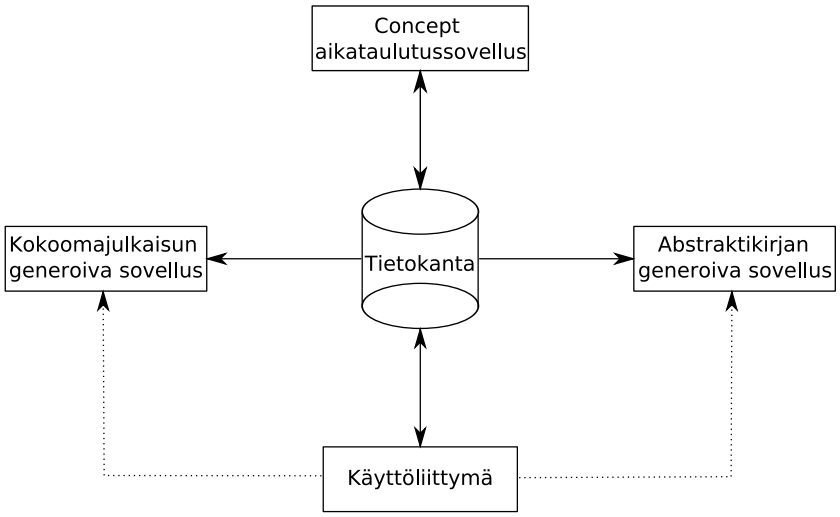
Kongressinhallintajärjestelmä koostuu tietokannasta ja useasta erillisestä sovelluksesta. Järjestelmä pohjautuu kokonaisuudessaan tietovarastoarkkitehtuuriin. Tietovarastoarkkitehtuurissa useat erilliset ohjelmat käyttävät ja päivittävät järjestelmän yhteisen tilan sisältävää tietovarastoa [18].

Kuvassa 2.1 on esitetty Kongressinhallintajärjestelmän rakenne ja suurimmat yksittäiset osat. Järjestelmän tietovarastona toimii PostgreSQL-tietokannanhallintajärjestelmässä [58] oleva tietokanta. Tietokanta sisältää kaiken kongressiin liittyvän tiedon lukuun ottamatta yksittäisten abstraktien tiedostoja, joihin tietokannassa on viitteet.

Kuvassa 2.1 käyttöliittymällä tarkoitetaan järjestelmän CGI-pohjaista käyttöliittymää eri käyttäjäryhmille. Käyttöliittymä on järjestelmän suurin yksittäinen osa, ja ainoa, jota muut kuin kongressin järjestäjät pääsevät käyttämään. Käyttöliittymä on toteutettu kokonaisuudessaan Perl-ohjelmointikielellä [52].

Käyttöliittymän avulla kongressin järjestäjä voi asettaa kongressin asetukset ja yleiset tiedot tietokantaan. Järjestäjä pääsee myös katselemaan erilaisia tilastotietoja sekä halutessaan muokkaamaan järjestelmään syötettyjä abstrakteja. Abstraktien syöttäminen, ja niiden arviointi, tehdään myös käyttöliittymän avulla.

Concept-aikataulutussovellus on Java-kielellä [21] toteutettu työasemasovellus,



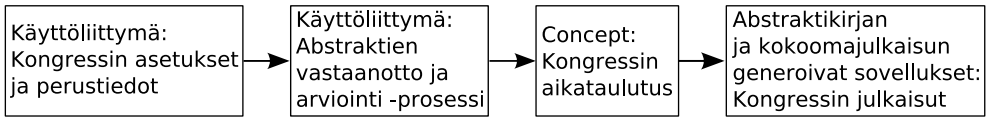
Kuva 2.1: Kongressinhallintajärjestelmän rakenne.

jolla luodaan kongressin aikataulu sekä suurin osa sessioista. Sovelluksella asetetaan myös kongressin esitykset sessioihin, ja lisätään sessiot aikatauluun. Toteutuksen ansiosta sovellusta voidaan käyttää sellaisenaan useassa eri käyttöjärjestelmässä.

Kongressinhallintajärjestelmä sisältää edellä mainittujen osien lisäksi kaksi erillistä kokonaisuutta kongressin julkaisujen tuottamiseen. Kokoomajulkaisun generoivalla sovelluksella tuotetaan kongressin kokoomajulkaisu HTML-sivuina. Kyseinen sovellus on toteutettu Perl-kielellä. Abstraktikirjan generoivalla sovelluksella ladotaan kongressin abstraktikirja ja yksittäisten abstraktien esikatseluversiot. Abstraktikirjan generoiva sovellus sisältää Perl-kielellä toteutetun kääreskriptin, joka käyttää Jyväskylän yliopiston informaatioteknologian tiedekunnan opinto-oppaan ladontaan kehitettyä ohjelmistoa. Kyseinen opinto-oppaan ladontaohjelmisto perustuu XML-tekniikoihin, ja käyttää pdf-tiedostojen latomiseen LaTeX-ladontaohjelmistoa.

Kongressin tieteellisten osien hallinta Kongressinhallintajärjestelmällä sisältää neljä erillistä osiota. Kuvassa 2.2 esitetään Kongressinhallintajärjestelmän osien suoritusjärjestys. Suoritusjärjestys ei ole käytännössä täysin lineaarinen, vaan eri sovelluksia voidaan käyttää, vaikka edellinen kongressin osio ei olisi vielä valmis.

Kongressin hallinnoinnin aluksi järjestelmään syötetään yleiset asetukset sekä tiedot kongressista, kuten kongressin aihealueet. Kun kongressin esivalmistelut on tehty, voidaan aloittaa abstraktien vastaanotto ja arviointi -prosessi. Tätä varten vaa-



Kuva 2.2: Kongressinhallintajärjestelmän osien suoritusjärjestys.

dittujen tietojen määrä vaihtelee suuresti kongressista riippuen. Esimerkiksi pienessä kongressissa järjestäjät voivat arvioida kaikki abstraktit, kun taas suuressa kongressissa arvioijia voi olla monia kymmeniä. Kongressinhallintajärjestelmä on suunniteltu mahdollisimman monenlaisten kongressien hallintaan.

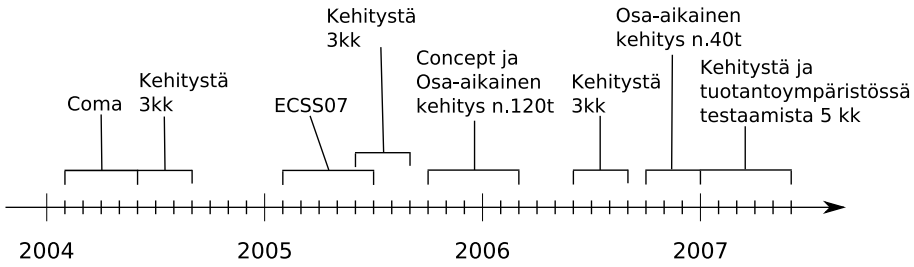
Abstraktien vastaanotto ja arviointi -prosessin aikana järjestelmään otetaan vastaan kongressin abstraktit. Tämä on järjestelmän toiminnan kannalta olennaisin osa, sillä sen aikana järjestelmää käyttävät useat ulkopuoliset käyttäjät. Abstraktien vastaanotto ja arviointi -prosessin toiminta antaa samalla kongressiin osallistuville ensimmäisen kuvan kongressin järjestelyiden laadusta.

Abstraktien vastaanottamisen ja arvioinnin jälkeen kongressin järjestäjät tekevät Concept-sovelluksella kongressin aikataulutuksen. Tämä sisältää kongressin sessioiden luonnin, esitysten jakamisen sessioihin, ja sessioiden asettamisen aikatauluun ja saleihin.

Kongressissa pidetyistä esityksistä tehdään erinäisiä julkaisuja kongressista riippuen. Kongressinhallintajärjestelmän avulla voidaan kongressin abstrakteista latoa painokelpoinen kirja pdf-muodossa. Lisäksi abstrakteista, niiden pidemmistä versioista, tai kokonaisista artikkeleista voidaan luoda CD-ROM:lla tai Internetissä julkaistava, erilaisilla hakutoiminnoilla varustettu, HTML-pohjainen kokoomajulkaisu. Kongressin lopulliset julkaisut tehdään vasta siinä vaiheessa, kun kaikki julkaisuihin tuleva tieto on lopullista. Esikatselua ja ulkoasun suunnittelua varten julkaisuja voidaan generoida myös aikaisemmin.

2.2 Kongressinhallintajärjestelmän kehitys

Kongressinhallintajärjestelmää on kehitetty vuoden 2004 alusta vuoden 2007 puoliväliin saakka. Järjestelmän kehitys on tapahtunut kolmessa Jyväskylän yliopiston tietotekniikan laitoksen opiskelijaprojektissa sekä erillisissä työjaksoissa noin puolentoista henkilötyövuoden verran. Kuvassa 2.3 esitetään Kongressinhallintajärjestelmän kehityksen aikajana.



Kuva 2.3: Kongressinhallintajärjestelmän kehityksen aikajana.

Kongressinhallintajärjestelmän kehitys alkoi helmikuussa 2004 Coma-opiskelijaprojektissa. Projektin tehtävänä oli tuottaa kesällä 2004 tietotekniikan laitoksella järjestettävän ECCOMAS2004-kongressin kokoomajulkaisu sähköisessä muodossa. Tästä syystä projekti keskittyi lähinnä Kongressinhallintajärjestelmän Kokoomajulkaisun generoivan sovelluksen kehittämiseen. Lisäksi projekti suunnitteli järjestelmän tietokannan perusrakenteen, ja kehitti muutaman ominaisuuden tietokannan tietojen käsittelyyn WWW-pohjaisella käyttöliittymällä.

Coma-projektissa kehitettyjä osia jatkokehitettiin kahden henkilön voimin kolmen kuukauden ajan kesällä 2004. Iso osa kesän 2004 töistä keskittyi ECCOMAS-kongressissa käytettyjen ohjelmien toteuttamiseen, joten tuona aikana vain järjestelmän tietokanta ja Kokoomajulkaisun generoiva sovellus kehittyivät Coma-projektin toteutuksista.

Kongressinhallintajärjestelmän käyttöliittymän kehitys jatkui keväällä 2005 olleessa ECSS07-opiskelijaprojektissa. Projektin aikana järjestelmää aloitettiin kehittämään kesällä 2007 Jyväskylän yliopiston liikuntabiologian laitoksella järjestettävää ECSS2007-kongressia varten. Projektin tehtävänä oli toteuttaa kongressin abstraktien vastaanotto ja arviointi -prosessi. Tämän lisäksi projekti kehitti järjestelmään käyttäjien tunnistuksen ja erinäisiä ominaisuuksia kongressin tietojen hallintaan, sekä jatkokehitti järjestelmän tietokantaa.

Kesän 2005 aikana koko järjestelmää jatkokehitettiin kolmen kuukauden ajan. Kesän aikana parannettiin sekä Coma että ECSS07 projektien toteutuksia, ja varmistettiin Kokoomajulkaisun generoivan sovelluksen toiminta muuttuneessa tietokannassa. Kesän 2005 aikana Kongressinhallintajärjestelmään lisättiin kokonaan uusi sovellus, Abstraktikirjan generoiva sovellus, jonka avulla kongressin abstraktikirjat, sekä yksittäisten abstraktien esikatseluversiot, voidaan taittaa ja generoida pdf-tiedostoina.

Syksystä 2005 kevääseen 2006 asti Kongressinhallintajärjestelmään kehitettiin uusi kokonaisuus Concept-opiskelijaprojektissa. Projekti suunnitteli ja toteutti kongressin aikataulutukseen Concept-sovelluksen. Samaan aikaan projektin kanssa Kongressinhallintajärjestelmän käyttöliittymään kehitettiin ECSS2007-kongressin vaatimusten mukaan uusia ominaisuuksia osa-aikatöinä.

Concept-projektin jälkeen Kongressinhallintajärjestelmän kehityksessä oli muutaman kuukauden tauko. Kesällä 2006 järjestelmän kehitys jatkui kolmen kuukauden ajan kahden henkilön voimin. Toinen kehittäjistä keskittyi Concept-sovelluksen jatkokehitykseen, ja toinen järjestelmän käyttöliittymän ja tietokannan jatkokehitykseen.

Kongressinhallintajärjestelmää on käytetty vuoden 2007 aikana kahden kongressin hallintaan. Näistä Jyväskylän yliopiston liikuntabiologian laitoksella kesällä 2007 järjestettävä ECSS2007-kongressi sisältää noin 1300 esitetystä, ja tietotekniikan laitoksella järjestettävä Eurogen2007-konferenssi noin 100 esitystä. Järjestelmän valmisteleminen kongressien tarpeisiin alkoi loppuvuodesta 2006. Vuoden 2007 aikana järjestelmään on kehitetty uusia ominaisuuksia kongressien järjestelyiden aikana huomattujen puutteiden mukaan. Samalla järjestelmää on päästy testaamaan oikeassa tuotantoympäristössä ja suurella käyttäjämäärällä. Testin aikana paljastuneet ohjelmavirheet on myös pystytty korjaamaan kevään 2007 aikana.

2.3 Abstraktien vastaanotto ja arviointi -prosessin logiikka

Kongressinhallintajärjestelmän WWW-pohjaisella käyttöliittymällä tehtävä abstraktien vastaanotto ja arviointi -prosessi on kongressin järjestelyiden olennaisin ja monimutkaisin osa. Järjestelmä sisältää neljä eri käyttäjäryhmää, erilaisilla oikeuksilla, jotka ovat tekemisissä prosessin kanssa:

- *Järjestäjillä* tarkoitetaan kongressin järjestäjiä, jotka käyttävät Kongressinhallintajärjestelmää.
- *Sessioiden järjestäjillä* tarkoitetaan yksittäisten sessioiden järjestäjiä. Suuret kongressit saattavat sisältää ulkoisesti hallittuja sessioita, joita sessioiden järjestäjät ovat järjestämässä.
- *Arvioijat* arvostelevat suuren osan kongressiin jätetyistä abstrakteista.
- *Yhteyshenkilöillä* tarkoitetaan henkilöitä, jotka jättävät kongressiin abstrakteja.

Ennen abstraktien vastaanotto ja arviointi -prosessin alkamista järjestelmään syötetään kongressin järjestäjät, jotka lisäävät mahdolliset sessioiden järjestäjät sekä aihealuekohtaiset arvioijat. Lisäksi kongressin aihealueet, sessiotyypit sekä mahdolliset erikoistapaukset syötetään järjestelmään ennen abstraktien vastaanottamisen aloittamista. Erikoistapauksilla tarkoitetaan kongressin sisäisiä tapahtumia, joihin abstrakti voi osallistua. Erikoistapaukselle voidaan määritellä myös arvioija, joka arvioi kaikki kyseiseen erikoistapaukseen liitetyt abstraktit. Esimerkiksi ECSS2007-kongressissa oli erikoistapauksena nuorten tutkijoiden kilpailu, johon jätetyt abstraktit asetettiin arvioitavaksi kilpailusta vastaaville.

Kongressin asetusten ja perustietojen asetuksen jälkeen voidaan abstraktien vastaanotto ja arviointi -prosessi aloittaa. Prosessin aikana abstraktit kiertävät järjestelmän sisällä eri tiloissa. Abstrakti voi olla yhdessä seuraavista kuudesta eri tilasta:

1. *Tallennettu*: Tilassa olevat abstraktit ovat vielä yhteyshenkilöllä syötettävinä. Yhteyshenkilö voi tallentaa abstraktin tiedot, ja palata myöhemmin muokkaamaan niitä. Mikäli tilassa olevaa abstraktia ei lähetetä arviointiin, sitä ei myöskään huomioida kongressin järjestelyissä mitenkään.

2. *Arvioitavana*: Tilassa olevat abstraktit on lähetetty kongressin arvioijille arvioitavaksi.

3. *Järjestäjällä*: Järjestäjillä olevat abstraktit odottavat järjestäjien toimenpiteitä. Järjestäjä voi esimerkiksi hyväksyä, hylätä tai asettaa abstraktin takaisin muokattavaksi.

4. *Muokattavana*: Tilassa olevat abstraktit ovat yhteyshenkilöiden muokattavana. Abstraktit sisältävät yleensä joitakin puutteita, ettei niitä voida hyväksyä sellaisenaan kongressiin ennen puutteiden korjaamista.

5. *Hylätty*: Tilassa olevat abstraktit ovat lopullisesti hylättyjä.

6. *Hyväksytty*: Kongressiin hyväksytyt abstraktit ovat tässä tilassa.

Abstraktien vastaanotto ja arviointi -prosessia on mallinnettu graafisesti väritetyillä Petri-verkoilla luvussa 5.2. Seuraavassa kuvataan sanallisesti prosessin logiikkaa.

Kenellä tahansa on oikeus pyytää abstraktilleen tunnukset järjestelmään. Kyseisestä henkilöstä tulee tällöin abstraktin yhteyshenkilö, ja abstrakti asetetaan tilaan

tallennettu. Tässä tilassa abstraktin yhteyshenkilöllä on oikeus muuttaa kaikkea abstraktista tallennettua tietoa, lukuun ottamatta abstraktin aihealuetta, joka valitaan pyydettäessä tunnukset abstraktille.

Tilaan *tallennettu* voidaan lisätä abstrakteja myös toista kautta. Nämä abstraktit ovat järjestäjien tai sessioiden järjestäjien tiettyyn sessioon erikseen kutsumia abstrakteja. Yhteyshenkilöllä on lähes samat oikeudet muokata sessioon kuuluvan abstraktin tietoja kuin itse hankkittujen tunnusten tapauksessa. Tällöin yhteyshenkilö ei kuitenkaan voi muuttaa abstraktin aihealuetta eikä sessiotyyppiä, jotka tulevat session mukaan. Sessioon kuuluvalle abstraktille ei voida myöskään asettaa erikoistapauksia.

Yhteyshenkilön jättäessä abstraktinsa arviointiin, voidaan se lähettää eri arvioijille, ja asettaa eri tilaan, abstraktin tiedoista riippuen. Ensimmäisenä tutkitaan kuuluuko abstrakti johonkin sessioon. Mikäli abstrakti kuuluu sessioon, se asetetaan kyseisen session järjestäjille arvioitavaksi, ja abstraktin tilaksi tulee *järjestäjällä*. Mikäli abstraktia ei ole asetettu sessioon, tutkitaan sen sessiotyyppiä. Sessiotyypin ollessa tyypiltään kutsuttuja abstrakteja varten, abstrakti asetetaan tilaan *järjestäjällä*, ja se jää odottamaan järjestäjien toimenpiteitä. Samoin tehdään myös siinä tapauksessa, että järjestelmässä ei ole asetettu *arvioijan valinta* asetusta päälle.

Abstrakteille, joita ei ole asetettu järjestäjille tai session järjestäjille arvioitavaksi, yritetään etsiä arvioijaa. Ensin tutkitaan onko abstrakti asetettu erikoistapaukseen, jolle on asetettu arvioija. Mikäli abstrakti kuuluu tällaiseen erikoistapaukseen, asetetaan abstrakti kyseiselle arvioijalle arvioitavaksi, ja tallennetaan tilaan *arvioitavana*. Muussa tapauksessa abstraktille etsitään seuraavaa aihealuekohtaista arvioijaa. Mikäli abstraktin aihealueelle on määrätty arvioijia, asetetaan abstrakti yhdelle heistä arviointiin, ja tallennetaan tilaan *arvioitavana*. Muussa tapauksessa abstrakti asetetaan tilaan *järjestäjällä*, ja se jää odottamaan järjestäjien toimenpiteitä.

Arvioijat voivat arvioida vain heille määrättyjä abstrakteja. Kongressin asetuksista, ja arvioijan antamasta arvosanasta riippuen, abstrakti asetetaan arvioinnin jälkeen joko tilaan *järjestäjällä* tai *hyväksytty*. Järjestäjä voi muuttaa päätöksellään abstraktien tilaa. Tilassa *järjestäjällä* olevat abstraktit odottavat järjestäjän päätöstä, mutta järjestäjillä on myös oikeus muuttaa missä tahansa tilassa olevien abstraktien tilaa. Järjestäjä voi hyväksyä tai hylätä abstraktin, sekä asettaa abstraktin muokattavaksi tai uudelle arvioijalle arvioitavaksi. Toimenpiteestä riippuen abstrakti siirretään joko tilaan *hyväksytty*, *hylätty*, *muokattavana* tai *arvioitavana*.

Abstraktin yhteyshenkilöllä on oikeus muokata tilassa *muokattavana* olevaa abstraktia. Abstraktin aihealuetta, sessiotyyppiä tai erikoistapauksia ei voida tällöin kuitenkaan muokata. Muutosten jälkeen yhteyshenkilö jättää muokattavana olleen abstraktin uudelleen arviointiin, jolloin se asetetaan edelliselle arvioijalle tai järjestäjille arvioitavaksi.

Tiloissa *hylätty* ja *hyväksytty* olevia abstrakteja ei yleensä enää siirretä toisiin tiloihin. Tarvittaessa järjestäjällä on kuitenkin oikeus tehdä näillekin abstrakteille uusi päätös. Järjestäjillä on lisäksi oikeus muokata kaikkien abstraktien kaikkia tietoja. Myös session järjestäjillä on oikeus muokata omaan sessioon kuuluvia abstrakteja, lukuun ottamatta niiden aihealuetta, sessiotyyppiä ja erikoistapauksia. Session järjestäjillä ei kuitenkaan ole oikeuksia muokata valmiiden sessioiden abstrakteja.

Yhteyshenkilöt voivat muokata omien abstraktiensa tietoja vain niiden ollessa tiloissa *tallennettu* tai *muokattavana*. Tällöinkin tietojen muokkauksessa on olemassa tiettyjä rajoituksia, kuten edellä tilasiirtymien yhteydessä on kuvattu. Kongressinhallintajärjestelmän arvioijilla ei sen sijaan ole minkäänlaisia abstraktien muokkausoikeuksia. He voivat vain jättää abstrakteista arvioita.

2.4 Tutkimusongelman esittely

Kongressinhallintajärjestelmä koostuu useista erillisistä osista, kuten luvussa 2.1 kuvattiin. Osista selvästi suurin ja kongressin järjestelyiden kannalta tärkein osio on luvussa 2.3 esitetty abstraktien vastaanotto ja arviointi -prosessi.

Kongressinhallintajärjestelmän kehitys on ollut pitkäaikaista, ja tapahtunut erinäisissä projekteissa, kuten luvussa 2.2 kuvattiin. Järjestelmään on kehitetty uusia ominaisuuksia lähinnä ilmenneiden tarpeiden mukaan. Aluksi järjestelmään toteutettiin vain Kokoomajulkaisun generoiva sovellus, mutta nykyisin Kongressinhallintajärjestelmä on kasvanut kattamaan kaikki tieteellisen kongressin hallinnassa vaaditut ominaisuudet. Tarpeiden mukaan tehtävä kehitys ilman pitkäjänteisempää kehityksen suunnittelua on kuitenkin riskialtista, sillä uudet ominaisuudet voivat huonossa tapauksessa vaikuttaa aikaisemmin kehitettyjen ominaisuuksien toimintaan.

Abstraktien vastaanotto ja arviointi -prosessi on Kongressinhallintajärjestelmän ainoa osa, jolla on kongressin järjestäjien ulkopuolisia käyttäjiä. Suuressa kongressissa käyttäjiä voi olla toista tuhatta. Nämä kaikki käyttäjät saavat ensimmäisen vaikutelman kongressin järjestelyiden laadukkuudesta Kongressinhallintajärjestelmän

WWW-pohjaisen käyttöliittymän kautta. Tästä syystä abstraktien vastaanotto ja arviointi -prosessin tulisi toimia mahdollisimman moitteettomasti, vähentäen näin järjestäjien työmäärää, ja parantaen kongressiin osallistuvien mielipidettä kongressin järjestelyistä.

Edellä esitellyistä syistä Kongressinhallintajärjestelmä sisältää useita mahdollisia tutkimuskohteita. Koko järjestelmän kannalta olennainen tutkimusongelma olisi Kongressinhallintajärjestelmän kaikkien osien saumaton yhteistoiminta. Abstraktien vastaanotto ja arviointi -prosessi sisältää myös usean mahdollisen tutkimuskohteen. ECSS07-projektista lähtien jatkuvasti kehitetty prosessin logiikka on tärkeä tutkimuskohde. Lisäksi prosessin tietoturva on tärkeää käyttöliittymän ollessa avoimena kaikille Internetin käyttäjille.

Kongressinhallintajärjestelmän eri osien yhteistoimintaa voidaan kuitenkin testata ECSS2007- ja Eurogen2007-kongressin yhteydessä. Yhteistoiminnassa mahdollisesti olevat virheet estävät todennäköisesti tiettyjen osien oikeellisen toiminnan, jolloin nämä virheet on järjestäjien myös helppo havaita. Abstraktien vastaanotto ja arviointi -prosessin tietoturva on puolestaan paljolti toteutusteknisistä yksityiskohdista riippuvaa, mikä tekee siitä järjestelmän suunnittelun kannalta vähemmän mielenkiintoisen tutkimusongelman. Tästä syystä tutkimuksen kohteeksi on valittu Kongressinhallintajärjestelmän abstraktien vastaanotto ja arviointi -prosessin logiikka.

Abstraktien vastaanotto ja arviointi -prosessin jatkuva kehitys on saanut kehittäjissä aikaan epäluuloa prosessin eri ominaisuuksien keskinäisestä toiminnasta. Prosessin eri ominaisuuksien vaikutuksia toisiinsa on vaikea testata järjestelmää käyttämällä, minkä takia prosessi on päätetty mallintaa, ja prosessin logiikkaa testata mallin avulla. Mallin analysoinnin tarkoituksena on vastata kysymykseen toimiiko abstraktien vastaanotto ja arviointi -prosessin suunnitelma toivotulla tavalla.

Tutkimuksen tarkoituksena on verifioida abstraktien vastaanotto ja arviointi -prosessin logiikka. Ohjelmiston verifioinnille on olemassa monia erilaisia määritelmiä. Pressmanin mukaan [59] verifiointi on joukko toimenpiteitä, jotka varmistavat, että ohjelmisto tekee oikein sille määritellyn toimenpiteen. Myös Boehmin yleisesti esitetty määritelmä verifioinnille [4], ”rakennammeko tuotetta oikein”, sopii tutkimuksen tapaukseen. Tutkimuksessa pyritään siis varmentamaan, ja mahdollisesti löytämään virheitä, Kongressinhallintajärjestelmän abstraktien vastaanotto ja arviointi -prosessin logiikan suunnitelmasta ja toteutuksesta.

3 Petri-verkot

Petri-verkot (Petri Nets) on formaali menetelmä diskreettien hajautettujen järjestelmien mallintamiseen ja analysointiin. Niillä on sekä havainnollinen graafinen esitystapa että vahva matemaattinen perusta. Itse Petri-verkot on yleisnimitys joukolle verkkoja, joille pätevät samat matemaattiset ja graafiset periaatteet.

Tässä luvussa annetaan yleiskuva Petri-verkoista ja esitellään niiden graafinen ja matemaattinen perusta sekä yleisimmät analysointimenetelmät ja kohteet. Petri-verkot esitellään käyttäen paikka-siirtymäverkkoja (Place/Transition Nets, P/T Nets), jotka yleisesti samaistetaan Petri-verkko käsitteen kanssa [62]. Luvussa esiteltävät käsitteet voidaan yleistää myös muille Petri-verkoille.

3.1 Johdatus Petri-verkkoihin

Luvussa esitellään Petri-verkkojen kehitys alkupäivistä nykyhetkeen. Tämän lisäksi annetaan yleiskuva Petri-verkkojen käyttökohteista.

3.1.1 Petri-verkkojen kehitys

Petri-verkot esitteli ensimmäisenä Carl Adam Petri väitöskirjassaan [55] (englanninkielinen käännös [56]). Petrin esittämä verkkomalli oli nimeltään ehto-tapahtuma-verkko (Condition/Event net). Tämä on yksinkertaisin Petri-verkko, ja se sisältää paljon rajoituksia verkon käsittelylle.

Peterson esittelee artikkelissaan [53] Petri-verkkojen varhaista historiaa. Petri-verkkojen kehitys alkoi laajeta kun väitöskirjassa esitetyt ideat tulivat yhdysvaltalaisen Applied Data Research Incin tutkijoiden tietoon. Kyseisessä tutkimusryhmässä kehitettiin Petri-verkkojen ensimmäiset teoriat ja esitysmuodot. Myöhemmin Petri-verkkojen tutkimus levisi MIT:n (Massachusetts Institute of Technology) tutkimusprojektiin, jossa Petri-verkkojen tutkimus jatkui tuotteliaana. Kyseinen tutkimusryhmä järjesti myös vuonna 1970 ja 1975 ensimmäiset Petri-verkkoja käsittelevät konferenssit.

1970-luvun lopulla Petri-verkkojen tutkimus levisi laajalle, ja varsinkin eurooppalaiset järjestivät lukuisia konferensseja Petri-verkkoihin liittyen. Kyseisten konfe-

renssien kokoomajulkaisut sisältävät monia Petri-verkkojen kannalta olennaisia artikkeleita. Vuodesta 1982 lähtien Springer-Verlag onkin julkaissut vuosittain *Applications and Theory of Petri Nets* -konferenssisarjan tärkeimpiä artikkeleita kirjoissaan *Advances in Petri Nets* ja *Applications and Theory of Petri Nets* sarjassa *Lecture Notes in Computer Science* [49]. Kyseisistä konferensseista kaksi on järjestetty Suomessa, vuonna 1985 Espoossa ja 2006 Turussa.

1970-luvulla paikka-siirtymäverkot olivat käytännön sovelluksissa käytetyin Petri-verkkomalli. Paikka-siirtymäverkon käyttäminen oikeiden järjestelmien mallintamisessa osoittautui kuitenkin ongelmalliseksi, sillä paikka-siirtymäverkon koko kasvaa erittäin suureksi siinä käytettyjen yksilöimättömien merkkien (token) takia. Ongelmasta johtuen eri aihealueisiin kehitettiin lukemattomia määriä variaatioita paikka-siirtymäverkoista, jotka yleisesti toimivat vain tietyissä erikoistapauksissa [32].

Ongelmaan kehitettiin ratkaisu kun predikaatti-tapahtumaverkot (Predicate / Transition nets, Pr/T nets) julkaistiin artikkelissa [19]. Predikaatti-tapahtumaverkko sisältää yksilöitäviä merkkejä, joiden siirtymäehdot noudattavat ensimmäisen asteen predikaattilogiikkaa. Kyseinen verkkomalli oli ensimmäinen niin sanottu korkean luokan Petri-verkko, joka oli kehitetty ilman erityistä sovellusaluekohdetta.

Predikaatti-tapahtumaverkkojen analysointi käyttäen paikka- ja siirtymäinvariantteja oli kuitenkin ongelmallista. Tätä korjaamaan kehitettiin ensimmäinen versio väritetyistä Petri-verkoista [27]. Tässä versiossa väritettyjen Petri-verkkojen kaarissa käytettiin funktioita, jotka eivät olleet niin kuvaavia kuin predikaatti-tapahtumaverkkojen käyttämä lausekemuoto. Tästä syystä kehitettiin uusi versio väritetyistä Petri-verkoista, jotka sisälsivät sekä predikaatti-tapahtumaverkkojen lausekemuodon että alkuperäisten väritettyjen Petri-verkkojen funtiomuodon [28]. Alun perin uuden verkkomallin nimi oli korkean luokan Petri-verkko (High-level Petri net), mutta sekaannusten välttämiseksi nimi muutettiin väritetyiksi Petri-verkoiksi (Coloured Petri Nets) [32].

Bernardinello ja Da Cindio esittelevät artikkelissa [2] Petri-verkkojen jaon kolmelle eri tasolle. Nämä ovat:

- **1-luokka**: Verkot, jotka voivat sisältää enintään yhden rakenteettoman merkin paikkaa kohti. Esimerkiksi ehto-tapahtumaverkko kuuluu tähän luokkaan.
- **2-luokka**: Verkot, jotka voivat sisältää useita rakenteettomia merkkejä jokaisessa paikassa. Esimerkiksi paikka-siirtymäverkko kuuluu tähän luokkaan.

- **3-luokka**: Korkean luokan Petri-verkot, jotka voivat sisältää rakenteellisia merkkejä paikoissa. Esimerkiksi väritetyt Petri-verkot ja predikaatti-tapahtumaverkot kuuluvat tähän luokkaan.

Nykyisin korkean luokan Petri-verkot ovat eniten käytössä niiden vaatiman tila-avaruuden pienuuden ja identifioitavien merkkien ansiosta. Korkean luokan Petri-verkoista on julkaistu standardi ISO/IEC 15909-1:2004 [25] vuoden 2004 lopulla. Standardi pohjautuu pitkälti väritettyihin Petri-verkkoihin. Lisäksi korkean tason Petri-verkkoja varten on kehitelty XML-pohjaista merkkausta nimeltä PNML (Petri Net Markup Language). Tämä työ on vielä kesken, mutta loppuvaiheessa.

3.1.2 Petri-verkkojen käyttökohteita

Alun perin Petri-verkot kehitettiin rinnakkaisuuden mallintamiseen ohjelmissa ja järjestelmissä. Petri-verkkojen luonteen takia ne soveltuvatkin tähän erinomaisesti, kuten myös erilaisten työvirtojen ja resurssienvarausjärjestelmien mallintamiseen. Näissä käyttökohteissa perinteisiä Petri-verkkoja on käytetty eniten. Kuitenkin viimeisen kolmen kymmenen vuoden aikana Petri-verkkoja on käytetty mallintamaan lukuisia erilaisia järjestelmiä ja systeemejä eri abstraktiotasoilla, ja niiden käyttäminen järjestelmien kehityksessä on ollut jatkuvasti kasvamaan päin.

Murata esittelee artikkelissaan [49] Petri-verkkojen onnistuneimmiksi sovelluskohteiksi tiedonsiirtoprotokollat ja suorituskyvyn arvioinnin. Protokollien kuulumisen onnistuneimpien sovelluskohteiden joukkoon on helppo ymmärtää niiden rinnakkaisuuden ja yhtäaikaisuuden takia. Courtiat ym. antavat artikkelissaan [9] ohjeita kuinka protokollia voidaan mallintaa ja analysoida Petri-verkkojen avulla. Esimerkiksi Petri-verkkojen mahdollistama protokollan validointi elävyyden (liveness), saavutettavuuden (reachability) ja rajoittuvuuden (boundedness) osalta on erittäin tärkeää. Petri-verkkojen analysointi esitetään tarkemmin luvuissa 3.3 ja 3.4.

Suorituskyvyn arviointiin Petri-verkot tarjoavat edullisen ja muunneltavan mallin. Valmiiksi mallinnettuun järjestelmään voidaan helposti lisätä suorituskykyä mittaavia ominaisuuksia korkean tason Petri-verkoissa, kuten väritetyissä Petri-verkoissa, sekä suorituskyvyn mittaamista varten kehitetyissä matalan tason Petri-verkoissa, kuten ajastetut Petri-verkot (Timed Petri Nets). Tällöin tulee käytettävän työkalun tietenkin myös tukea suorituskyvyn mittaamista.

Artikkeleissa [49] ja [39] esitellään myös muita sovelluskohteita, joissa Petri-verkkoja on käytetty menestyksellä. Näistä tärkeimpiä ovat aikaisemmin mainittu-

jen lisäksi käyttöjärjestelmät, rinnakkaiset ohjelmat, hajautetut järjestelmät, perinteisen teollisuuden tuotannonohjausjärjestelmät sekä tietovuo-ohjelmat. Periaatteessa Petri-verkkoja voidaan siis käyttää kaikenlaisten rinnakkaisuutta ja tietovuota sisältävien järjestelmien mallintamiseen ja analysointiin niin ohjelmistotekniikan kuin perinteisen teollisuudenkin parissa.

Petri-verkkojen käyttö teollisuudessa on kasvanut viimeisten vuosien aikana riittävän pitkälle kehittyneiden korkean tason Petri-verkkojen ja näitä varten kehitettyjen työkalujen ansiosta. Verkkosivulla [16] listataan noin 60 projektia väritettyjen Petri-verkkojen käytöstä, joista jokaisesta on julkaistu vähintään yksi artikkeli. Osa näistä projekteista on toteutettu yritysmaailmassa, kuten Nokialla, Ericssonilla ja Hewlett-Packardilla.

3.2 Petri-verkkojen peruskäsitteet ja graafinen esitysmuoto

Petri-verkko on suunnattu graafi, joka sisältää kahdenlaisia solmuja ja näiden välisiä kaaria. Petri-verkon olennaisimpia ominaisuuksia on se, että sitä voidaan suorittaa kuten ohjelmaa. Tämän ansiosta Petri-verkosta voidaan analysoida staattisten ominaisuuksien lisäksi dynaamisia ominaisuuksia. Näitä esitellään tarkemmin luvussa 3.3. Tässä luvussa määritellään Petri-verkkojen matemaattiset peruskäsitteet sekä graafinen esitysmuoto.

3.2.1 Petri-verkon määrittely

Määritellään seuraavaksi Petri-verkot, kuten ne on määritelty paikka-siirtymäverkoille.

Määritelmä 3.1 *Petri-verkko on kuusikko* (S, T, F, K, M_0, W) *[60], missä*

- S *on paikkojen joukko.*
- T *on siirtymien joukko.*
- $F \subseteq (S \times T) \cup (T \times S)$ *on suunnatun verkon kaarien joukko. Määritelmästä nähdään, että Petri-verkko voi sisältää kaaria vain paikasta siirtymään tai toisin päin. Paikkojen (siirtymien) välillä ei siis voi olla kaarta.*
- $K : S \to \mathbb{N} \cup \{\infty\}$ *on paikkojen kapasiteettifunktio. Funktio määrää jokaiselle paikalle sen maksimimäärän merkkejä. Paikan merkkien maksimimäärä voi olla myös ääretön, jolloin paikalla ei ole maksimimäärää merkkejä.*

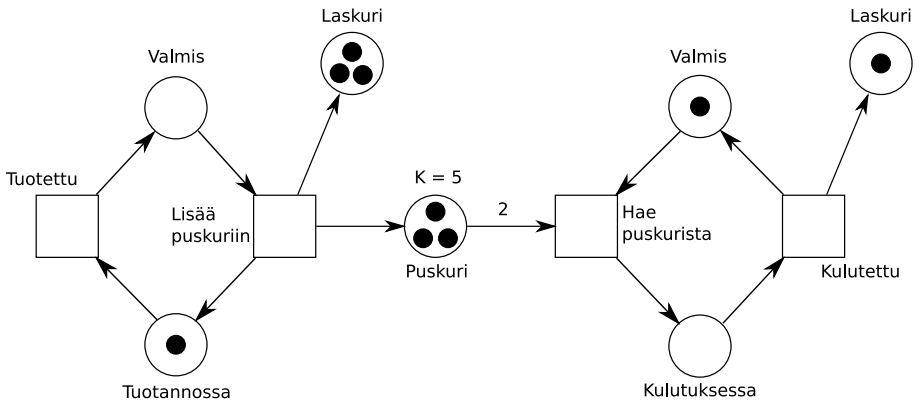
- $M_0 : S \to \mathbb{N} \cup \{\infty\}$ *on verkon alkumerkkaus. Se on funktio, joka määrää alkutilassa verkon jokaisen paikan sisältävien merkkien lukumäärän. Alkumerkkaukselle täytyy päteä* $M_0(s) \leq K(s), \forall s \in S$.

- $W : F \to \mathbb{N} \setminus \{0\}$ *on kaarien painojen funktio. Funktio määrittelee jokaiselle verkon kaarelle painon, joka on luonnollinen luku pois lukien 0.*

Lisäksi täytyy päteä $S \cap T = \emptyset$.

Kaikki perinteiset Petri-verkot sisältyvät määritelmään 3.1 mukautettujen rajoitusten kanssa. Esimerkiksi 1-luokkaan kuuluvissa Petri-verkoissa, kuten ehto-tapahtumaverkko, kapasiteettifunktio saa arvon yksi kaikissa paikoissa, ja kaarien painot ovat yksi kaikilla kaarilla. Tällaisessa tapauksessa kapasiteettifunktio ja verkon kaarien painot jätetään määritelmästä pois.

Petri-verkon graafisessa esityksessä paikat kuvataan ellipseinä tai ympyröinä, ja siirtymät suorakulmioina tai neliöinä. Tavallisissa Petri-verkoissa merkkejä kuvataan mustilla ympyröillä. Lisäksi kaarien painot merkitään kaarien yhteyteen ja paikan kapasiteetti paikan yhteyteen. Kaarille ei tavallisesti merkitä painoa mikäli se on yksi, eikä paikoille kapasiteettia mikäli se on ääretön. Kuvassa 3.1 on esitetty esimerkkikuva Petri-verkosta.



Kuva 3.1: Esimerkki tuottaja-kuluttaja järjestelmästä paikka-siirtymäverkkona.

Kuvassa 3.1 esitetty tuottaja-kuluttaja järjestelmä esittää perinteisten Petri-verkkojen graafisen esitysmuodon. Kuva on muokattu lähteen [60] esimerkistä. Kuvasta nähdään ympyröillä piirretyt paikat ja neliöillä piirretyt siirtymät. Näiden välillä on suunnatut kaaret. Kaikkien kaarien paino, yhtä lukuun ottamatta, on yksi, sillä näihin ei ole erikseen merkitty kaaren painoa. Puskurin kapasiteetti on viisi, muiden

paikkojen ääretön. Kuvaan on lisäksi merkitty verkon merkkaus mustilla pienillä ympyröillä. Kuvassa olevilla paikkojen ja siirtymien nimillä ei ole verkon koneellisen analysoinnin kannalta merkitystä, mutta ihmiselle nimet kuvaavat verkkoa olennaisesti, kuten hyvin nimetyt muuttujat ohjelmakoodissa.

3.2.2 Petri-verkon suorittaminen

Petri-verkon tärkein ominaisuus on sen dynaamisuus, eli sitä voidaan suorittaa kuten ohjelmaa. Petri-verkon tilaa voidaan muuntaa suorittamalla (firing) siirtymä, joka muuntaa verkon merkkausta. Määritellään verkon mahdollista siirtymää ja tulevaa käyttöä varten siirtymän ja paikan esi- ja jälkijoukot.

Määritelmä 3.2 *Siirtymän ja paikan esi- ja jälkijoukot ovat:*

- ${}^*t = \{s \mid (s,t) \in F\}$ *on siirtymän t esijoukko. Siirtymän esijoukko on niiden paikkojen joukko, joista tulee kaari kyseiseen siirtymään.*
- $t^* = \{s \mid (t,s) \in F\}$ *on siirtymän t jälkijoukko. Siirtymän jälkijoukko on niiden paikkojen joukko, joihin siirtymästä on kaari.*
- ${}^*s = \{t \mid (t,s) \in F\}$ *on paikan s esijoukko. Paikan esijoukko on niiden siirtymien joukko, joista tulee kaari kyseiseen paikkaan.*
- $s^* = \{t \mid (s,t) \in F\}$ *on paikan s jälkijoukko. Paikan jälkijoukko on niiden siirtymien joukko, joihin paikasta on kaari.*

Nyt voidaan määritellä verkon mahdollinen siirtymä.

Määritelmä 3.3 *Verkon siirtymä t on mahdollinen (enabled) [60], mikäli pätee:*

1. $\forall s \in {}^*t : M(s) \geq W(s,t)$
2. $\forall s \in t^* : M(s) \leq K(s) - W(t,s)$

Määritelmän 3.3 kohta 1 määrää, että siirtymän jokaisen esipaikan tulee sisältää vähintään paikan ja siirtymän välisen kaaren painon verran merkkejä. Jälkimmäinen määrittelee, että siirtymän minkään jälkipaikan merkkien lukumäärä ei saa ylittää paikan kapasiteettia suoritettaessa siirtymä.

Siirtymän suorittaminen muuntaa verkon merkkausta. Siirtymän t tapahtuminen merkkauksessa M_0 muuntaa merkkauksen M_0 merkkaukseksi M_1, ja tätä merkitään $M_0[t\rangle M_1$. Tämä voidaan yleistää myös siirtymien tapahtumasarjaan (occurrence sequence), jolloin merkkauksesta M_0 päästään merkkaukseen M_n siirtymien tapahtumasarjalla $M_0[t_1\rangle M_1[t_2\rangle M_2 \ldots M_{n-1}[t_n\rangle M_n$. Suoritettaessa siirtymä, Petri-verkon merkkaus muuttuu määritelmän 3.4 mukaan.

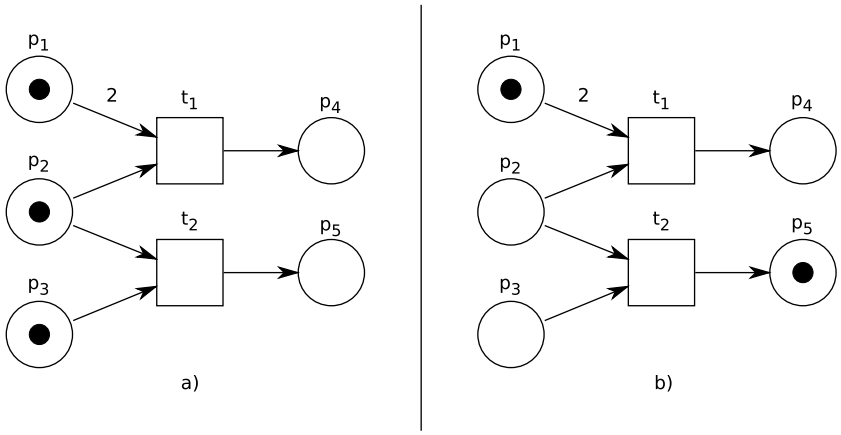
Määritelmä 3.4 *Petri-verkon merkkauksen M_0 muutos $\forall s \in S$ suoritettaessa siirtymä t:*

$$M_1(s) = \begin{cases} M_0(s) - W(s,t), & \textit{mikäli } s \in {}^*t \setminus t^* \\ M_0(s) + W(t,s), & \textit{mikäli } s \in t^* \setminus {}^*t \\ M_0(s) - W(s,t) + W(t,s), & \textit{mikäli } s \in {}^*t \cap t^* \\ M_0(s), & \textit{muussa tapauksessa} \end{cases}$$

Määritelmän 3.4 mukaan paikan merkkien lukumäärä vähenee paikan ja siirtymän kaaren painon verran, mikäli paikka kuuluu siirtymän esijoukkoon, mutta ei jälkijoukkoon. Mikäli paikka kuuluu siirtymän jälkijoukkoon, mutta ei esijoukkoon, sen merkkien lukumäärään lisätään siirtymän ja paikan välisen kaaren painon verran merkkejä. Lisäksi mikäli paikka kuuluu sekä siirtymän esijoukkoon että jälkijoukkoon, sovelletaan molempia sääntöjä, siis paikasta vähennetään pois lähtevän kaaren painon verran merkkejä ja lisätään tulevan kaaren painon verran merkkejä. Muussa tapauksessa paikka ei liity siirtymään mitenkään, ja sen merkkien lukumäärä pysyy samana. Kuvassa 3.2 esitellään kuinka merkkaus muuttuu suoritettaessa siirtymä.

Kuvasta 3.2 a) nähdään, että alkuperäisessä merkkauksessa vain siirtymä t_2 on mahdollinen otettaessa huomioon määritelmässä 3.3 esitetyt säännöt. Siirtymä t_1 ei ole mahdollinen, sillä kaikissa sen esijoukon alkioissa ei ole riittävää määrää merkkejä. Siirtymän t_2 suorituksen jälkeen merkkaus muuttuu kuvan 3.2 b) mukaiseksi. Paikoille määritellään uudet merkkien lukumäärät määritelmässä 3.4 esitettyjen sääntöjen mukaan. Paikkojen p_1 ja p_4 merkkaus pysyy täysin samana, sillä ne eivät kuulu siirtymän t_2 esi- tai jälkijoukkoon. Sen sijaan paikoista p_2 ja p_3 poistetaan yksi merkki, ja paikkaan p_5 lisätään yksi merkki, sillä näiden kaikkien, ja siirtymän t_2 välisen kaaren paino on yksi. Petri-verkon suoritus ei siis välttämättä säilytä merkkien lukumäärää koko verkossa.

Petri-verkon siirtymiä voidaan suorittaa yhtäaikaisesti (concurrently). Tällöin tulee varmistua siitä, että suoritettavat siirtymät eivät ole keskenään ristiriidassa.



Kuva 3.2: Esimerkki merkkauksen muuttumisesta suoritettaessa siirtymä. a) merkkaus ennen siirtymän suorittamista, b) merkkaus siirtymän t_2 suorituksen jälkeen.

Yleisesti kaksi siirtymää t_1 ja t_2 voidaan suorittaa yhtäaikaisesti, mikäli ei ole väliä suoritetaanko siirtymät järjestyksessä t_1t_2 vai t_2t_1.

Olkoon $U \subseteq T, U \neq \emptyset$ joukko yhtäaikaisesti suoritettavia siirtymiä. Tällöin voidaan määritellä yhtäaikaisesti suoritettaville siirtymille kaksi erilaista siirtymäsääntöä. Heikko siirtymäsääntö (weak transition rule) määritellään samaan tapaan kuin määritelmissä 3.3 ja 3.4 määriteltiin yhdelle siirtymälle. Näin yhtäaikaisesti suoritettavien siirtymien joukko $t_1, t_2, \ldots, t_n \in U$ voidaan suorittaa yhtäaikaisesti $M_0[U\rangle M_n$ sen sijaan, että ne suoritettaisiin sarjassa [26].

Vahva siirtymäsääntö (strict transition rule) määrittelee tarkemman säännön yhtäaikaisesti suoritettaville siirtymille.

Määritelmä 3.5 *Vahvan siirtymäsäännön mukaan joukko siirtymiä U voidaan suorittaa yhtäaikaisesti, mikäli*

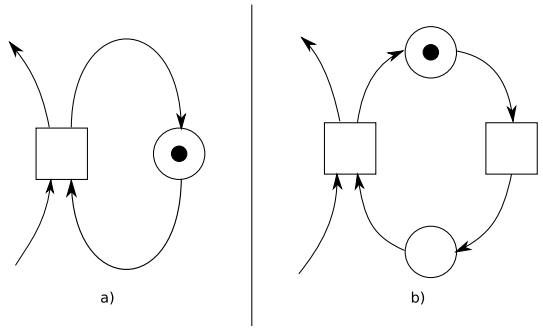
$$\forall t_1, t_2 \in U : t_1 \neq t_2 \Rightarrow ({}^*t_1 \cup t_1^*) \cap ({}^*t_2 \cup t_2^*) = \emptyset$$

on tosi [26].

3.2.3 Petri-verkon matriisimuoto

Petri-verkko voidaan esittää matriisimuodossa mikäli verkko on puhdas (pure). Petri-verkko on puhdas mikäli siinä ei ole yhtään itseissilmukkaa (self-loop). Itseissilmukalla tarkoitetaan sellaista silmukkaa, missä paikka kuuluu siirtymän sekä esi- että jälkijoukkoon. Epäpuhtaista Petri-verkoista voidaan tehdä puhtaita lisäämällä

itseissilmukkaan ylimääräinen paikka ja siirtymä [49]. Kuvassa 3.3 on esitetty itseissilmukan sisältämä epäpuhdas Petri-verkko ja sen muuntaminen puhtaaksi Petri-verkoksi poistamalla itseissilmukka.



Kuva 3.3: a) Epäpuhdas Petri-verkko, joka sisältää itseissilmukan. b) Muunnettu akohdan verkko puhtaaksi Petri-verkoksi poistamalla itseissilmukka.

Puhdas Petri-verkko voidaan esittää matriisimuodossa esittämällä se kytkentämatriisilla (Incidence matrix). Kytkentämatriisi on $n \times m$ -matriisi, missä n on paikkojen ja m siirtymien lukumäärä. Matriisin alkiot muodostetaan siten, että alkio

$$C_{s,t} = W(t,s) - W(s,t) \text{ [26].}$$

Matriisin alkio (s,t) on siirtymästä t paikkaan s tulevan kaaren paino vähennettynä paikasta s siirtymään t menevän kaaren paino. Tästä syystä kytkentämatriisin vaatimuksena on puhdas Petri-verkko, sillä puhtaassa verkossa ei ole kaarta samasta paikasta siirtymään ja toisin päin. Mikäli paikan ja siirtymän välillä ei ole ollenkaan kaarta, on kyseisen kytkentämatriisin alkion arvo 0. Esimerkkinä esitetään kuvan 3.2 Petri-verkon kytkentämatriisi.

$$\mathbf{C} = \begin{bmatrix} -2 & 0 \\ -1 & -1 \\ 0 & -1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Verkon merkkaus M esitetään sarakevektorina, joka sisältää alkion jokaista paikkaa kohti. Alkion arvo on kyseisen paikan sisältämien merkkien lukumäärä. Suoritettavia siirtymiä kuvataan vektorilla $\vec{u}$. Se on sarakevektori, joka sisältää alkion jokaista siirtymää varten. Vektorin alkion arvo kuvaa kuinka monta kertaa kyseinen

siirtymä suoritetaan. Suoritettaessa yksi siirtymä, on vektorissa $\vec{u}$ vain yksi alkio arvoltaan 1, ja muiden arvo on 0 [26]. Määritellään seuraavaksi kuinka Petri-verkon merkkauksen muutos voidaan laskea.

Määritelmä 3.6 *Petri-verkon merkkauksen muutos voidaan laskea kytkentämatriisin ja siirtymävektorin kertolaskuna*

$$M' = M + C\vec{u},$$

missä M' on verkon uusi merkkaus, M lähtömerkkaus ja $\vec{u}$ suoritettavat siirtymät.

Esimerkkinä uuden merkkauksen laskemisesta esitetään kuvan 3.2 siirtymän t_2 suoritus käyttämällä edellä esitettyä kytkentämatriisia.

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -2 & 0 \\ -1 & -1 \\ 0 & -1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Esimerkistä nähdään kuinka alkuperäinen merkkaus $(1,1,1,0,0)^T$ sisälsi yhden merkin paikoissa p_1, p_2 ja p_3. Tähän lisätään kytkentämatriisin ja suoritettavien siirtymien vektorin kertolasku. Esimerkissä suoritetaan siirtymä t_2 yhden kerran, minkä takia siirtymävektori on $(0,1)^T$. Tuloksena saadaan Petri-verkon uusi merkkaus $(1,0,0,0,1)^T$, eli paikoissa p_1 ja p_5 on yksi merkki.

3.3 Petri-verkoista analysoitavat ominaisuudet

Suunniteltavasta järjestelmästä mallinnetusta Petri-verkosta on graafisen esitysmuotonsa ansiosta hyötyä järjestelmän suunnittelijoille sen tarjoaman havainnollisen kuvauksen ansiosta. Graafisesta esitysmuodosta nähdään samalla kertaa sekä järjestelmän rakenne että toiminta. Petri-verkkojen selvästi tärkein ominaisuus on kuitenkin mahdollisuus niiden koneelliseen analysointiin.

Petri-verkoista voidaan analysoida kahdenlaisia ominaisuuksia, dynaamisia (behavioral properties) ja rakenteellisia (structural properties). Dynaamiset ominaisuudet riippuvat verkon alkumerkkauksesta toisin kuin rakenteelliset ominaisuudet, joille ei verkon merkkauksella ole merkitystä, vaan pelkästään paikkojen ja siirtymien keskinäisellä rakenteella [49]. Rakenteellisia ominaisuuksia voidaan yleensä analysoida kytkentämatriisin ja tästä muodostettujen yhtälöiden avulla. Sen sijaan

dynaamisia ominaisuuksia analysoitaessa täytyy monesti käyttää muita menetelmiä, sillä kytkentämatriisi ei kuvaa verkon sen hetkistä tilaa millään lailla.

Laajojen ja kompleksisten järjestelmien ominaisuuksien täydellinen analysointi on hankalaa Petri-verkon mahdollisesti äärettömän tila-avaruuden takia. Tällaisissa tapauksissa voidaan rajoittaa mallinnetun verkon analysointia vain osaan verkon paikoista, ja tutkimalla näiden ominaisuuksia. Tietyissä tapauksissa Petri-verkko voidaan myös koostaa hierarkkisesti pienemmistä osista, jolloin laajakin verkko voidaan analysoida täydellisesti analysoimalla sen yksittäisiä osia.

Petri-verkoista voidaan analysoida lukuisia erilaisia dynaamisia ja rakenteellisia ominaisuuksia. Näistä yleisimmät analysointikohteet ovat saavutettavuus (reachability), elävyys (liveness) ja rajoittuvuus (boundedness). Seuraavassa esitellään edellä mainittujen ominaisuuksien lisäksi kotitila (home state) ja oikeudenmukaisuus (fairness). Näissä keskitytään lähinnä dynaamisiin ominaisuuksiin.

3.3.1 Saavutettavuus

Saavutettavuudella tutkitaan onko mallinnetun järjestelmän mahdollista saavuttaa tietty tila. Järjestelmän saavutettavien tilojen tutkiminen on olennainen osa minkä tahansa suunniteltavan järjestelmän analysointia. Petri-verkkojen tapauksessa tutkitaan voiko mallinnettu verkko saavuttaa tietyn merkkauksen lähtien alkumerkkauksesta. Tietyissä tapauksissa on mielekkäämpää tarkastella koko Petri-verkon merkkauksen sijaan sen aliverkon merkkausta. Tällöin tutkitaan vain tiettyjen paikkojen merkkauksia, jolloin puhutaan aliverkon saavutettavuudesta (submarking reachability).

Saavutettavuutta voidaan tarkastella tutkimalla onko olemassa tapahtumasarja $\sigma = t_0, t_1, \ldots, t_n$, eli sarja suoritettavia siirtymiä siten, että päästään alkumerkkauksesta M_0 merkkaukseen M_n. Tätä merkitään $M_0[\sigma\rangle M_n$. Mikäli tällainen tapahtumasarja on olemassa, on merkkaus M_n saavutettavissa. Petri-verkon N kaikkia saavutettavia tiloja alkumerkkauksesta M_0 merkitään $R(N, M_0)$, jolloin saavutettavuuden tutkimisessa tutkitaan päteekö $M_n \in R(N, M_0)$ [49].

Saavutettavuuden tutkiminen on ratkaistavissa oleva ongelma, mikä on todistettu Kosarajun artikkelissa [38]. Yleisessä tapauksessa saavutettavuuden tutkiminen vaatii kuitenkin eksponentiaalisen tilan ja ajan [49]. Saavutettavuuden tutkimiseen kehitettyjä menetelmiä on useita ja näistä esitetään muutama luvussa 3.4.

3.3.2 Elävyys

Elävyydellä tutkitaan järjestelmän suorituksen jatkumista. Elävä järjestelmä ei joudu missään vaiheessa deadlock-tilaan, eli sellaiseen tilaan, josta suoritusta ei voida enää jatkaa. Tämä vaatimus on erittäin olennainen monille eri sovellusalueille, kuten tietoliikenneprotokollille [9] ja käyttöjärjestelmille. Sen sijaan kertasuoritteisissa ohjelmissa elävyyttä ei tarvita, sillä ohjelman suorituksen tuleekin päättyä tarvittavien toimintojen loputtua.

Petri-verkko on elävä mikäli kaikista saavutettavista merkkauksista on olemassa sarja siirtymiä siten, että mikä tahansa siirtymä voidaan suorittaa. Elävän Petri-verkon kaikki siirtymät voidaan siis suorittaa äärettömän monta kertaa.

Koko Petri-verkon elävyys ei ole kaikissa tapauksissa vaadittava ominaisuus. Tällöin voidaan tutkia myös yksittäisten siirtymien elävyyttä. Murata määrittelee artikkelissaan [49] siirtymille viisi eri elävyyden tasoa.

Määritelmä 3.7 *Siirtymän elävyyden tasot. Siirtymä t on:*

0) L0-elävä (kuollut), mikäli ei ole olemassa tapahtumasarjaa σ siten, että siirtymä t suoritettaisiin.

1) L1-elävä, mikäli siirtymä t suoritetaan vähintään kerran jossain tapahtumasarjassa σ.

2) L2-elävä, mikäli siirtymä t suoritetaan vähintään k kertaa jossain tapahtumasarjassa σ.

3) L3-elävä, mikäli siirtymä t suoritetaan äärettömän monta kertaa jossain tapahtumasarjassa σ.

4) L4-elävä tai elävä, mikäli siirtymä on L1-elävä $\forall M \in R(N, M_0)$.

Yksittäisten siirtymien avulla voidaan määritellä myös Petri-verkon elävyys.

Määritelmä 3.8 *Petri-verkon sanotaan olevan luokan Lk-elävä, mikäli $\forall t \in T : t$ on Lk-elävä.*

Näin ollen Petri-verkko on elävä mikäli kaikki sen siirtymät ovat luokan L4-eläviä. Petri-verkolle voidaan määritellä myös rakenteellinen elävyys (structural liveness). Petri-verkon sanotaan olevan rakenteellisesti elävä, mikäli on olemassa jokin alkumerkkaus M_0 siten, että Petri-verkko on elävä.

3.3.3 Rajoittuvuus

Petri-verkon rajoittuvuudella tarkoitetaan sen paikkojen sisältämien merkkien yläarajaa.

Määritelmä 3.9 *Petri-verkon sanotaan olevan k-rajoitettu mikäli*

$$\forall s \in S, \forall M \in R(N, M_0) : M(s) \leq k.$$

k-rajoitetun Petri-verkon yksikään paikka ei saa sisältää missään saavutettavassa merkkauksessa yli k kappaletta merkkejä [49]. Petri-verkon rajoittuvuuden sijaan tutkitaan yleensä yksittäisten paikkojen rajoittuvuutta, sillä tämä antaa yksittäisistä paikoista paljon tarkempaa tietoa. Koko Petri-verkon ylärajaksi tulee sen eniten merkkejä sisältävän paikan yläraja, mutta tutkittaessa yksittäisiä paikkoja, voidaan näille saada paljon pienempiä ylärajoja. Esimerkiksi kuvan 3.1 Petri-verkko ei ole rajoitettu, sillä molemmat laskuri-paikat ovat rajoittamattomia. Sen sijaan muille verkon paikoille voidaan määrittää ylärajat.

Paikkojen ylärajoja voidaan soveltaa suoraan käytännössä. Esimerkiksi järjestelmän erilaisten puskureiden yläraja voidaan todentaa tutkimalla rajoittuvuutta, ja näin estää vaaralliset puskureiden ylivuodot.

Rajoittamattoman Petri-verkon täydellinen analysointi ilman erityisiä menetelmiä on mahdotonta, sillä rajoittamaton Petri-verkko voi sisältää äärettömän määrän merkkauksia. Tätä varten on kehitelty erilaisia menetelmiä, joita esitellään luvussa 3.4.

Petri-verkolle on määritelty myös rakenteellinen rajoittuvuus (structural boundedness). Petri-verkon sanotaan olevan rakenteellisesti rajoitettu mikäli se on rajoitettu millä tahansa äärellisellä alkumerkkauksella [49].

3.3.4 Kotitila

Monissa järjestelmissä on tavoitteena päästä suorituksen päättyessä takaisin johonkin tilaan. Esimerkiksi palvelimen, joka vastaanottaa yhteyden ja tekee erilaisia toimintoja ennen yhteyden sulkemista, tulee jossain vaiheessa päätyä takaisin tilaan, jossa se voi jälleen vastaanottaa uusia yhteyksiä.

Määritelmä 3.10 *Kotitila (Home State) on merkkaus M' siten, että $\forall M \in R(N, M_0)$, M' on saavutettavissa.*

Kotitila on Petri-verkon sellainen merkkaus, joka voidaan saavuttaa kaikista verkon saavutettavista tiloista. Peruutettavuudella (reversibility) tarkoitetaan sitä, että Petri-verkon alkumerkkaus M_0 on kotitila.

3.3.5 Reiluus

Petri-verkon reiluusominaisuuksia (fairness properties) on määritelty useita erilaisia, ja niitä käytetään järjestelmän erilaisten ominaisuuksien tutkimiseen. Yleisesti tutkitaan joko siirtymien reiluutta tai Petri-verkon merkkauksien reiluutta. Tässä käydään läpi vain siirtymien reiluusominaisuuksia. Merkkauksille voidaan määritellä reiluus samaan tapaan.

Carstensen ja Valk määrittelevät artikkelissaan [5] rajoitetut reiluusominaisuudet. Joukko siirtymiä ovat keskenään *rajoitetusti reiluja (bounded-fair, B-fair)* mikäli yhtään siirtymistä ei voida suorittaa tiettyä ylärajaa useammin ilman, että myös muut siirtymät suoritettaisiin. Petri-verkon sanotaan olevan rajoitetusti reilu mikäli verkon kaikki siirtymät ovat keskenään rajoitetusti reiluja.

Petri-verkon tapahtumasarjan σ avulla voidaan määritellä reiluusominaisuuksia. Lehmann, Pnueli ja Stavi määrittelevät artikkelissaan [43] kolme erilaista reiluuden tasoa.

Määritelmä 3.11 *Olkoon $U \subseteq T$ joukko Petri-verkon siirtymiä ja σ tapahtumasarja. Tapahtumasarja σ on*

- *tasapuolinen (impartial), mikäli σ on äärellinen tai jokainen siirtymä $u \in U$ suoritetaan äärettömän monta kertaa.*
- *reilu (fair), mikäli σ on äärellinen tai jokainen siirtymä $u \in U$, joka on mahdollinen äärettömän monta kertaa tapahtumasarjassa σ myös suoritetaan äärettömän monesti.*
- *oikeudenmukainen (just), mikäli σ on äärellinen tai jokainen siirtymä $u \in U$, joka on jostain merkkauksesta lähtien jatkuvasti mahdollinen myös suoritetaan äärettömän monta kertaa.*

Tapahtumasarjalle edellä määritellyt reiluuden tasot voidaan yleistää koko Petri-verkolle. Petri-verkon sanotaan olevan tasapuolinen mikäli $U = T$ ja jokainen tilassa $M \in R(N, M_0)$ mahdollinen tapahtumasarja σ on tasapuolinen. Samaan tapaan määritellään Petri-verkon oikeudenmukaisuus ja reiluus.

Petri-verkon reiluusominaisuuksia tutkitaan monesti analysoitaessa eläviä järjestelmiä. Tällöin järjestelmän suoritus on periaatteessa ikuista ja reiluutta voidaan tutkia äärettömien tapahtumasarjojen avulla. Analysoitavasta järjestelmästä riippuen siirtymillä halutaan olevan tietyn tasoista tai ei ollenkaan reiluutta.

3.4 Petri-verkkojen analysointimenetelmät

Petri-verkoille on kehitetty lukuisia erilaisia analysointimenetelmiä. Nämä jakautuvat kolmeen eri luokkaan: kattavuuspuuta käyttäviin menetelmiin, tilayhtälöä käyttäviin menetelmiin sekä paikka- ja siirtymäinvariantteja käyttäviin menetelmiin. Luvussa esitetään pääpiirteittäin kaikki edellä mainitut menetelmät.

3.4.1 Kattavuuspuu

Kattavuuspuu (coverability tree, coverability graph) on yleisin menetelmä Petri-verkkojen analysointiin. Se soveltuu teoriassa kaikenlaisten Petri-verkkojen analysoimiseen. Käytännössä ongelmaksi voi tulla kattavuuspuun kasvaminen liian isoksi tehokasta analysointia varten. Tämä johtuu laajojen ja monimutkaisten Petri-verkkojen tilojen suuresta määrästä (state space explosion). Petri-verkon erilaisten tilojen määrää voidaan rajoittaa erilaisilla menetelmillä. Näitä esitellään väritetyille Petri-verkoille luvussa 4.4.3.

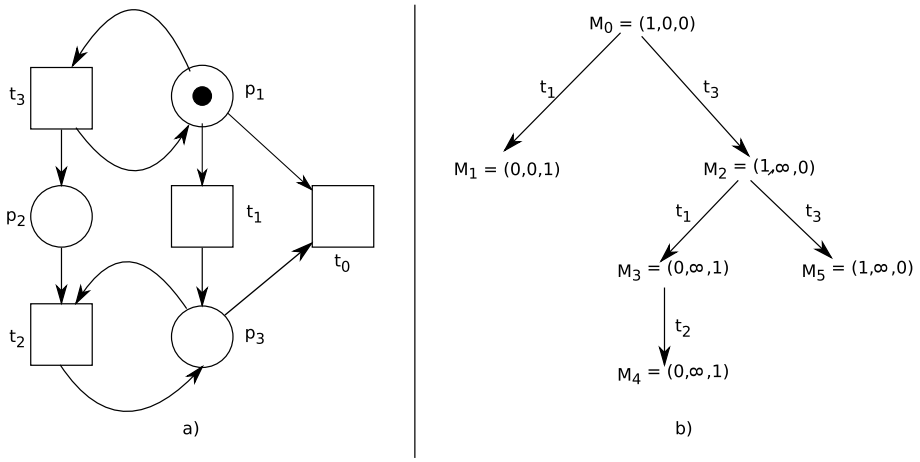
Kattavuuspuun esittelemistä varten otetaan käyttöön ∞-merkki, jolle pätee säännöt: $\forall n \in \mathbb{N} : \infty > n, \infty \pm n = \infty, \infty \geq \infty$. Kattavuuspuun muodostaminen tehdään seuraavalla algoritmilla, joka on mukailtu lähteen [49] esittämästä algoritmista.

1. Aseta Petri-verkon alkumerkkaus M_0 puun juurisolmuksi ja merkkaa se käsittelemättömäksi solmuksi.
2. Niin kauan kun puussa on käsittelemättömiä solmuja tee:
   (a) Valitse jokin käsittelemätön solmu. Solmun merkkaus olkoon M.
   (b) Mikäli merkkaus M on jo olemassa polulla solmusta juurisolmuun, merkkaa solmu käsitellyksi ja palaa kohtaan 2.
   (c) Niin kauan kun merkkauksessa M on suorittamattomia mahdollisia siirtymiä:
      i. Suorita siirtymä t, jonka seurauksena saadaan merkkaus M'.

ii. Mikäli polulla merkkauksesta M juurisolmuun on olemassa solmu M'' siten, että $\forall s \in S : M'(s) \geq M''(s)$ ja $M' \neq M''$, on merkkaus M'' jo katettu merkkauksella M'. Tällöin merkkaus M' korvataan merkkauksella, missä $\forall s \in S : M'(s) > M''(s)$, $M'(s) = \infty$.

iii. Asetetaan merkkaus M' puuhun uudeksi solmuksi solmun M lapseksi, ja merkitään se käsittelemättömäksi solmuksi.

Kuvassa 3.4 esitetään yksinkertainen Petri-verkko ja sen kattavuuspuu [49]. Kuvan alkumerkkauksessa voidaan suorittaa kaksi siirtymää, t_1 ja t_3. Suoritettaessa siirtymä t_1 päädytään merkkaukseen $M_1 = (0, 0, 1)$, eikä yksikään siirtymistä ole enää mahdollinen. Suoritettaessa siirtymä t_3 päädytään merkkaukseen $M_2 = (1, 1, 0)$, ja huomataan, että merkkauksesta M_2 polulla juureen on solmu M_0, jossa $\forall s \in S : M_2(s) \geq M_0(s)$. Näin korvataan merkkaus $M_2 = (1, 1, 0)$ merkkauksella $M_2 = (1, \infty, 0)$, sillä $M_2(p_2) > M_0(p_2)$. Jatkettaessa samalla tavalla saadaan kuvassa esitetty kattavuuspuu.



Kuva 3.4: Esimerkki Petri-verkosta ja sen kattavuuspuusta. a) Petri-verkko b) Petri-verkon kattavuuspuu.

Rajoitetulla Petri-verkolla on äärellinen määrä erilaisia merkkauksia, ja siten myös kattavuuspuu sisältää äärellisen määrän merkkauksia. ∞-merkin käyttöönottaminen kattavuuspuussa mahdollistaa myös äärettömän Petri-verkon esittämisen äärellisessä kattavuuspuussa. Lisäksi kattavuuspuu sisältää jokaisen saavutettavissa olevan merkkauksen tai sen kattavan merkkauksen. Kirjassa [60] Reisig esittää todistuksen sille, että kattavuuspuu on aina äärellinen ja sisältää kaikki Petri-verkon merkkaukset tai niiden kattavuudet.

Kattavuuspuun avulla voidaan tutkia useita luvussa 3.3 esitettyjä Petri-verkon dynaamisia ominaisuuksia. Kaikki luvussa esitetyt ominaisuudet voidaan ratkaista kattavuuspuun avulla mikäli Petri-verkko on rajoitettu. Rajoittamattoman Petri-verkon tapauksessa joudutaan usein käyttämään muitakin menetelmiä [49].

Petri-verkko on rajoitettu, mikäli kattavuuspuun yksikään solmu ei sisällä ∞-merkkiä. Rajoitetun Petri-verkon tapauksessa kattavuuspuusta käytetään nimitystä saavutettavuuspuu (reachability tree), sillä saavutettavuuspuun solmut sisältävät suoraan kaikki rajoitetun Petri-verkon saavutettavat merkkaukset. Näin ollen saavutettavuuspuun avulla voidaan ratkaista Petri-verkon saavutettavuus- ja rajoittuvuusominaisuudet. Saavutettavuuspuusta voidaan myös tutkia Petri-verkon elävyys ja kotitilaominaisuudet, sillä nämä ovat verrattavissa saavutettavuuden ongelmaan.

Kattavuuspuun analysointi on ongelmallisempaa mikäli Petri-verkko ei ole rajoitettu. Tällöin voidaan tutkia yksittäisten paikkojen rajoittuvuuksia, vaikka koko Petri-verkko ei olekaan rajoitettu. Kattavuuspuun sisältämien solmujen suurin arvo paikalle on myös kyseisen paikan yläraja. Mikäli kattavuuspuu sisältää jossakin solmussa paikan kohdalla ∞-merkin, kyseinen paikka ei ole rajoitettu [14].

Muista Petri-verkon ominaisuuksista ei voida olla täysin varmoja, sillä rajoittamattoman Petri-verkon kattavuuspuussa käytetty ∞-merkki hävittää tietoa verkon tiloista. Esimerkiksi kuvassa 3.4 esitetystä kattavuuspuusta ei voida sanoa varmuudella onko merkkaus $(1, 5, 0)^T$ saavutettavissa. Mikäli verkon kaarien painot olisivat yhdestä poikkeavia, kyseinen merkkaus ei välttämättä olisi saavutettavissa.

Rajoittamattoman Petri-verkon kattavuuspuusta ei voida myöskään tutkia verkon elävyyden tasoa. Murata esittelee artikkelissaan [49] kaksi Petri-verkkoa, joilla on täysin sama kattavuuspuu, mutta joista toinen on elävä ja toinen ei-elävä. Sen sijaan kuolleet siirtymät voidaan löytää kattavuuspuun avulla.

Rajoittamattoman Petri-verkon kotitilaominaisuus on ratkaistavissa mikäli voidaan ratkaista onko kotitila saavutettavissa. Tämä voidaan varmistaa siten, että löydetään kattavuuspuusta solmu, joka vastaa kotitilaa ilman ∞-merkkiä. Täydellisesti kotitilaominaisuuttakaan ei siis voida ratkaista kattavuuspuun avulla mikäli kyseessä on rajoittamaton Petri-verkko.

3.4.2 Tilayhtälö

Petri-verkon dynaamisia ominaisuuksia voidaan tutkia ilman koko verkon saavutettavien tilojen etsimistä. Tähän käytetään Petri-verkon kytkentämatriisia ja tilasiir-

tymiä, kuten luvussa 3.2.3 on esitetty. Näitä lineaarista algebraa hyödyntäviä menetelmiä käytetään vain erikoistapauksissa, esimerkiksi tutkittaessa onko tietty yksittäinen tila saavutettavissa.

Murata [48] määrittelee Petri-verkoille määritelmään 3.6 perustuvan tilayhtälön:

$$M_k = M_{k-1} + C\vec{u}_k, k = 1, 2, \ldots$$

missä Petri-verkon tila M_{k-1} muunnetaan tilaksi M_k kertomalla kytkentämatriisi siirtymävektorilla $\vec{u}_k$.

Tilayhtälön avulla voidaan tutkia onko Petri-verkon tietty tila saavutettavissa. Petri-verkon tila M_d on saavutettavissa alkutilasta M_0, mikäli on olemassa sarja tilasiirtymiä, joilla päästään alkutilasta M_0 tilaan M_d:

$$M_d = M_0 + C \sum_{k=1}^{d} \vec{u}_k.$$

Merkitsemällä $\Delta M = M_d - M_0$ ja $\vec{x} = \sum_{k=1}^{d} \vec{u}_k$, voidaan tilayhtälö esittää muodossa $\Delta M = C\vec{x}$. Tästä riittää tutkia yhtälön toteuttavia mahdollisia vektorin $\vec{x}$ epänegatiivisia ratkaisuja.

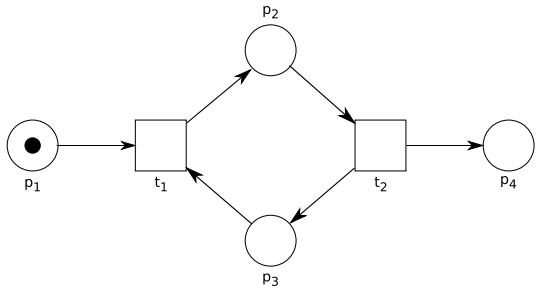
Edellä esitetyn tilayhtälön toteuttavat vektorin $\vec{x}$ ratkaisut eivät välttämättä ole mahdollisia. Peterson antaa kirjassaan [54] tästä esimerkin. Kuvan 3.5 Petri-verkon alkumerkkaus $M_0 = (1, 0, 0, 0)^T$ ja kytkentämatriisi

$$C = \begin{bmatrix} -1 & 0 \\ 1 & -1 \\ -1 & 1 \\ 0 & 1 \end{bmatrix}$$

Tutkittaessa onko merkkaus $M_d = (0, 0, 0, 1)^T$ saavutettavissa alkumerkkauksesta saadaan yhtälölle

$$\Delta M = C\vec{x} \Rightarrow \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 1 & -1 \\ -1 & 1 \\ 0 & 1 \end{bmatrix} \cdot \vec{x}$$

ratkaisuksi $\vec{x} = (1, 1)^T$, joka kuvaa tapahtumasarjoja $t_1 t_2$ ja $t_2 t_1$. Kuvasta 3.5 kuitenkin nähdään, ettei kumpikaan siirtymistä ole mahdollinen alkumerkkauksessa



Kuva 3.5: Esimerkki Petri-verkosta, jonka tilayhtälöön on olemassa ratkaisu, joka ei ole mahdollinen.

M_0. Näin ollen tilayhtälön ratkaisun löytyminen ei takaa kaikissa tapauksissa, että tutkittava tila olisi saavutettavissa mahdollisilla siirtymillä. Sen sijaan ratkaisun puuttuminen takaa, että kyseinen tila ei ole saavutettavissa.

Murata esittää artikkelissaan [48] toisenlaisen menetelmän saavutettavuusongelman ratkaisemiseen. Menetelmän avulla voidaan tutkia onko tietty tila saavutettavissa alkutilasta ottamatta kantaa tähän vaadittavista siirtymistä, toisin kuin etsittäessä ratkaisua vektorille $\vec{x}$. Menetelmä on tietyissä tapauksissa tehokkaampi tapa ratkaista saavutettavuusongelma, kuin etsittäessä ratkaisua vektorille $\vec{x}$, sillä ratkaisun $\vec{x}$ etsiminen voi olla erittäin hankalaa tai sitä ei edes ole olemassa.

Olkoon p tutkittavan Petri-verkon paikkojen lukumäärä, t siirtymien lukumäärä ja r kytkentämatriisin aste (rank). Kytkentämatriisin aste voidaan ratkaista esimerkiksi Gaussin eliminointimenetelmällä [17], jota ei tässä esitellä tarkemmin. Menetelmässä oletetaan, että ehto $r < p$ pätee. Mikäli $r = p$, Petri-verkon sanotaan olevan *täydellisesti saavutettavissa (completely reachable)*, jolloin mistä tahansa alkutilasta voidaan saavuttaa kaikki muut tilat [48].

Tiedettäessä p ja r, voidaan kytkentämatriisin transpoosi jakaa osiin seuraavasti:

$$C^T = \begin{bmatrix} C_{11}^T & C_{12}^T \\ C_{21}^T & C_{22}^T \end{bmatrix}$$

missä C_{11}^T on $r \times (p-r)$ -matriisi, C_{12}^T on $r \times r$ -matriisi, C_{21}^T on $(t-r) \times (p-r)$ -matriisi, ja C_{22}^T on $(t-r) \times r$ -matriisi. Lisäksi matriisi C_{12}^T on ei-singulaarinen eli sille on olemassa käänteismatriisi.

Tilayhtälölle $C\vec{x} = \Delta M$ on olemassa ratkaisu $\vec{x}$ ainoastaan mikäli ΔM on ortogonaalinen tilayhtälön homogeenisen systeemin $C^T\vec{y} = \vec{0}$ jokaisen ratkaisun $\vec{y}$ suhteen. Olkoon I $(p-r) \times (p-r)$ yksikkömatriisi. Yksikkömatriisin I ja aikaisemmin

määriteltyjen matriisien C_{11} ja C_{12} avulla voidaan muodostaa matriisi B_f, jonka rivit ovat yhtälön $C^T\vec{y} = \vec{0}$ lineaarisesti riippumattomia ratkaisuja $\vec{y}$. Saadaan $(p - r) \times p$ -matriisi

$$B_f = \left[I : -C_{11}(C_{12})^{-1} \right]$$

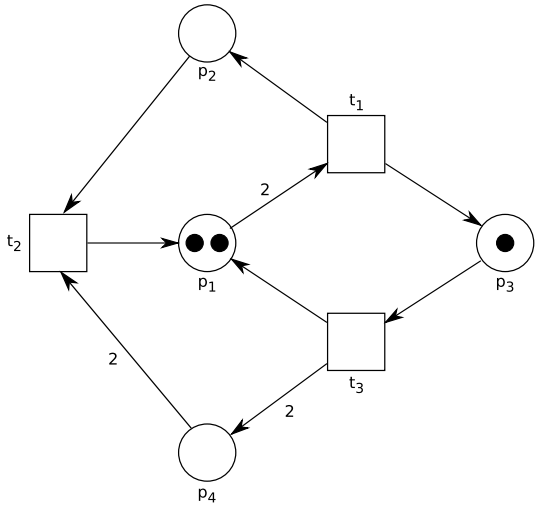
Tilayhtälön homogeenisen systeemin $C^T\vec{y} = \vec{0}$ avulla havaitaan, että $C^T B_f^T = 0$ eli matriisit C ja B_f ovat ortogonaalisia keskenään. Aikaisemmin määriteltiin, että tilayhtälölle on olemassa ratkaisu $\vec{x}$ ainoastaan mikäli ΔM on ortogonaalinen tilayhtälön homogeenisen systeemin jokaisen ratkaisun $\vec{y}$ suhteen. Matriisin B_f rivit antavat vektorin $\vec{y}$ ratkaisut, mistä seuraa, että tilayhtälölle on olemassa ratkaisu ainoastaan mikäli yhtälö

$$B_f \Delta M = \vec{0}$$

pätee.

Tämäkään menetelmä ei takaa, että tila M_d olisi saavutettavissa tilasta M_0 mahdollisilla siirtymillä muuta kuin erikoistapauksessa. Mikäli tutkittava Petri-verkko on elävä merkattu graafi (Marked Graph), antaa menetelmä täydellisen ratkaisun saavutettavuusongelmaan. *Merkattu graafi* on Petri-verkkojen sellainen aliluokka, jonka jokaiseen paikkaan tulee ja lähtee vain yksi kaari. Tarkempaa tietoa merkatuista graafeista ja niiden analysoimisesta on luettavissa Muratan artikkelista [47].

Analysoidaan esimerkkinä Muratan menetelmästä kuvassa 3.6 esitettyä Petri-verkkoa. Kuvan Petri-verkon alkumerkkaus $M_0 = (2, 0, 1, 0)^T$.



Kuva 3.6: Muratan menetelmän esimerkin Petri-verkko.

Kuvan 3.6 Petri-verkon kytkentämatriisin transpoosi on

$$C^T = \begin{bmatrix} -2 & 1 & 1 & 0 \\ 1 & -1 & 0 & -2 \\ 1 & 0 & -1 & 2 \end{bmatrix}.$$

Petri-verkko sisältää neljä paikkaa ja kytkentämatriisin aste on kaksi, joten $r < p$. Näiden tietojen avulla voidaan kytkentämatriisin transpoosista muodostaa matriisit C_{11} ja C_{12} sekä jälkimmäisen käänteismatriisi C_{12}^{-1}:

$$C_{11} = \begin{bmatrix} -2 & 1 \\ 1 & -1 \end{bmatrix}, C_{12} = \begin{bmatrix} 1 & 0 \\ 0 & -2 \end{bmatrix}, C_{12}^{-1} = \begin{bmatrix} 1 & 0 \\ 0 & -\frac{1}{2} \end{bmatrix}$$

Olkoon I 2×2 yksikkömatriisi. Nyt voidaan muodostaa matriisi B_f. Saadaan

$$B_f = \left[I : -C_{11}(C_{12})^{-1} \right] = \begin{bmatrix} 1 & 0 & 2 & \frac{1}{2} \\ 0 & 1 & -1 & -\frac{1}{2} \end{bmatrix}$$

Tutkitaan nyt matriisin B_f avulla voidaanko merkkaus $M_d = (3,0,0,2)^T$ saavuttaa. Tällöin $\Delta M = (1,0,-1,2)^T$, mistä seuraa:

$$B_f \Delta M = \begin{bmatrix} 1 & 0 & 2 & \frac{1}{2} \\ 0 & 1 & -1 & -\frac{1}{2} \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ -1 \\ 2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \vec{0}$$

Yhtälön ratkaisu osoittaa, että merkkaus $M_d = (3,0,0,2)^T$ on mahdollista saavuttaa. Ratkaisu ei kuitenkaan takaa sitä, että merkkaus M_d saavutetaan mahdollisilla siirtymillä.

Tutkitaan seuraavaksi onko merkkaus $M_d = (1,1,1,0)^T$ saavutettavissa. Tällöin $\Delta M = (-1,1,0,0)^T$, mistä saadaan yhtälön ratkaisuksi $B_f \Delta M = (-1,1)^T \neq \vec{0}$. Tämä todistaa, että merkkaus $M_d = (1,1,1,0)^T$ ei ole saavutettavissa alkutilasta $M_0 = (2,0,1,0)^T$.

Aikaisemmin todettiin, että Muratan menetelmän avulla ei voida todistaa merkkauksen saavutettavuutta kuin erikoistapauksessa. Esimerkkinä tästä voidaan käyttää kuvan 3.5 Petri-verkkoa. Verkon alkumerkkaus $M_0 = (1,0,0,0)^T$. Tutkittaessa onko merkkaus $(0,0,0,1)^T$ saavutettavissa lähtömerkkauksesta M_0 saadaan $\Delta M = (-1,0,0,1)^T$. Kyseisestä Petri-verkosta muodostettu matriisi B_f on

$$B_f = \begin{bmatrix} 1 & 0 & -1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

Tämän avulla saadaan yhtälön ratkaisuksi $B_f \Delta M = \vec{0}$. Alkumerkkauksesta M_0 ei kuitenkaan voida suorittaa yhtään siirtymää, joten merkkaukseen M_d ei ole olemassa mahdollista tapahtumasarjaa. Näin ollen tilaa M_d ei voida myöskään saavuttaa.

3.4.3 Paikka- ja siirtymäinvariantit

Paikkainvariantti (S-invariant) kuvaa joukkoa paikkoja, joiden merkkien painotettu lukumäärä pysyy samana kaikissa saavutettavissa olevissa merkkauksissa. *Siirtymäinvariantti (T-invariant)* kuvaa kuinka monta kertaa kukin Petri-verkon siirtymä tulee suorittaa jostakin lähtömerkkauksesta, jotta päädytään takaisin tähän merkkaukseen.

Edellä esitetystä paikka- ja siirtymäinvarianttien kuvauksesta voidaan huomata niiden käyttötarkoitus järjestelmien analysoinnissa. Paikkainvariantit sopivat lähes kaikenlaisten järjestelmien analysoimiseen. Niiden avulla voidaan analysoida esimerkiksi elävyys- ja rajoittuvuusominaisuuksia, sillä paikkainvarianteista voidaan johtaa kaikissa Petri-verkon tiloissa päteviä paikkojen merkkauksia kuvaavia yhtälöitä. Siirtymäinvarianteilla voidaan analysoida esimerkiksi järjestelmän kotitilaominaisuuksia. Tämän ansiosta ne ovat erittäin hyödyllisiä varsinkin erilaisten tilallisten järjestelmien, kuten tilallisten protokollien, analysoinnissa.

Paikka- ja siirtymäinvariantit ovat vektoreita, ja niiden määrittelyssä käytetään apuna Petri-verkon kytkentämatriisia. Olkoon $|S|$ Petri-verkon paikkojen ja $|T|$ siirtymien lukumäärä.

Määritelmä 3.12 *Olkoon $\vec{x}$ $|S|$-alkioinen vektori. Tällöin paikkainvariantti voidaan määritellä kahdella eri tavalla [60, 49]:*

- *$\vec{x}$ on paikkainvariantti, mikäli $C^T\vec{x} = \vec{0}$.*
- *Mikäli $\forall M_n \in R(N, M_0)$ pätee $M_n^T\vec{x} = M_0^T\vec{x}$, on $\vec{x}$ paikkainvariantti.*

Määritelmä 3.13 *Olkoon $\vec{y}$ $|T|$-alkioinen vektori. Tällöin siirtymäinvariantti voidaan määritellä kahdella eri tavalla [60, 49]:*

- *$\vec{y}$ on siirtymäinvariantti, mikäli $C\vec{y} = \vec{0}$.*
- *Mikäli on olemassa sarja siirtymiä σ siten, että $M_0[\sigma\rangle M_0$ ja vektori $\vec{y}$, joka sisältää täsmälleen sarjan σ siirtymien lukumäärät, on $\vec{y}$ siirtymäinvariantti.*

Paikka- ja siirtymäinvariantit voidaan määritellä rakenteellisesti kytkentämatriisin avulla tai dynaamisesti käyttämällä Petri-verkon saavutettavien tilojen joukkoa. Molemmissa tapauksissa paikka- ja siirtymäinvariantit ovat vektoreita. Paikkainvariantin alkiot kuvaavat kyseisen paikan merkkien painoarvoa, ja siirtymäinvariantin alkiot kuvaavat kuinka monta kertaa kyseinen siirtymä tulee suorittaa.

Rakenteellinen määritelmä pätee kaikille Petri-verkon paikka- ja siirtymäinvarianteille. Siirtymäinvarianttien tapauksessa ongelmia aiheuttaa se, että kaikki siirtymäinvariantit eivät ole mahdollisia. Tästä syystä dynaaminen määritelmä siirtymäinvarianteille on rakenteellista parempi, sillä tällöin löydetään vain mahdollisia siirtymäinvariantteja. Myös paikkainvarianttien dynaaminen määritelmä on hyödyllinen sen mahdollistaessa eri merkkausten välisten yhtälöiden muodostamisen.

Paikka- ja siirtymäinvarianttien tapauksessa ongelmia on aiheuttanut niiden löytäminen. Lähes kaikille Petri-verkoille voidaan löytää lukemattomia paikka- ja siirtymäinvariantteja etsimällä edellä esitettyjen yhtälöiden toteuttavia vektoreita. Käytännössä Petri-verkot sisältävät tuhansia paikkoja ja siirtymiä, jolloin yhtälöiden ratkaiseminen tulee työlääksi, eivätkä löydetyt invariantit ole välttämättä järjestelmän toiminnan kannalta olennaisia.

Jensen [32] toteaa, että paras tietämys järjestelmässä olevista invarianteista on sen suunnittelijalla. Mallintaessaan Petri-verkkoa, suunnittelija tietää – tai voi löytää – järjestelmän spesifikaatioista paikkoja ja siirtymiä, joiden tulisi muodostaa analysoinnin kannalta olennaisia invariantteja. Suunniteltujen invarianttien avulla Petri-verkko voidaan mallintaa, ja myöhemmin analysoida siten, että kyseiset invariantit pätevät.

Invariantteja voidaan muodostaa lisää jo olemassa olevista invarianteista. Määritellään seuraavassa muutama yksinkertainen tapaa muodostaa uusia invariantteja.

Määritelmä 3.14 *Paikka- ja siirtymäinvarianteille pätee [60]*

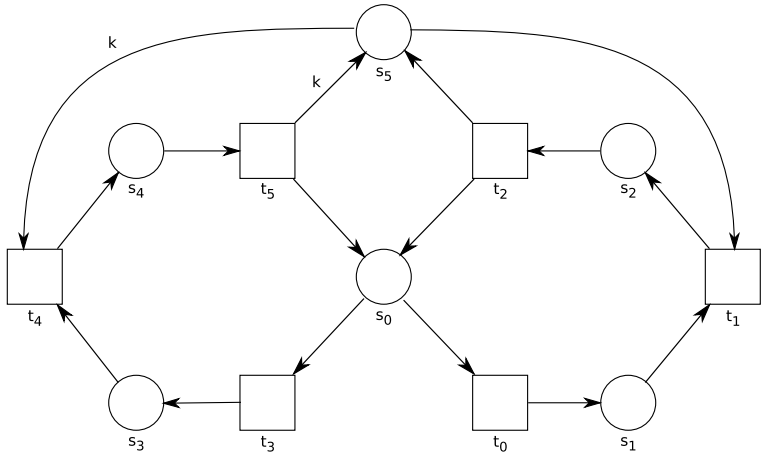
- *Olkoon $\vec{i}_1$ ja $\vec{i}_2$ paikkainvariantteja sekä $z \in \mathbb{Z}$ kokonaisluku. Tällöin $\vec{i}_1 + \vec{i}_2$ ja $z\vec{i}_1$ ovat myös paikkainvariantteja.*
- *Olkoon $\vec{i}_1$ ja $\vec{i}_2$ siirtymäinvariantteja sekä $z \in \mathbb{Z}$ kokonaisluku. Tällöin $\vec{i}_1 + \vec{i}_2$ ja $z\vec{i}_1$ ovat myös siirtymäinvariatteja.*

Molemmat edellä esitetyt ominaisuudet ovat selviä, kun muistetaan miten paikka- ja siirtymäinvariantit määriteltiin määritelmissä 3.12 ja 3.13. Paikkainvarianttien ominaisuus mahdollistaa useampi paikkaisten invarianttien muodostamisen järjestel-

män analysointia varten. Sen sijaan siirtymäinvarianttien yhdistäminen ei takaa sitä, että uusi siirtymäinvariantti olisi mahdollinen edes silloin kun molemmat yhdistettävät siirtymäinvariantit olisivat mahdollisia erikseen.

Esimerkkinä käytetään kuvassa 3.7 esitettyä käyttöjärjestelmän kuvitteellisen puskurin lukevia ja kirjoittavia prosesseja, sekä niiden hallintaa mallintavaa Petri-verkkoa. Alussa paikoissa s_0 ja s_5 on k kappaletta merkkejä, ja muut paikat ovat tyhjiä. Puskuria voi lukea samaan aikaan enintään k kappaletta prosesseja tai siihen voi kirjoittaa yksi prosessi. Lukevia ja kirjoittavia prosesseja ei voi olla samaan aikaan. Kuvassa 3.7 esitetyn Petri-verkon kytkentämatriisi on

$$C = \begin{bmatrix} -1 & 0 & 1 & -1 & 0 & 1 \\ 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & -1 & 1 & 0 & -k & k \end{bmatrix}$$



Kuva 3.7: Käyttöjärjestelmän puskurin luku- ja kirjoitusprosessit [60]. Merkit paikoissa kuvaavat seuraavaa: s_0: epäaktiiviset prosessit, s_1: lukemaan valmiit prosessit, s_2: lukevat prosessit, s_3: kirjoittamaan valmiit prosessit, s_4: kirjoittavat prosessit, s_5: prosessien hallinta.

Petri-verkolle voidaan löytää seuraavat järjestelmän kannalta olennaiset paikkainvariantit: $i_1 = (1, 1, 1, 1, 1, 0)^T$ ja $i_2 = (0, 0, 1, 0, k, 1)^T$. Molemmat täyttävät edellä esitetyn paikkainvariantin määritelmän. Ensimmäinen paikkainvariantti i_1 kuvaa

sitä, kuinka prosessit jakautuvat eri tiloihin s_0–s_4. Toinen paikkainvariantti i_2 kuvaa kirjoittavien ja lukevien prosessien suhdetta.

Paikkainvariantin i_1 avulla voidaan muodostaa kaikille saavutettaville merkkauksille $M_n \in R(N, M_0)$ seuraava yhtälö [60]:

$$\sum_{i=0}^{4} M_0(s_i) = \sum_{i=0}^{4} M_n(s_i) = n$$

Yhtälö kuvaa sitä, että kaikissa saavutettavissa tiloissa prosessien lukumäärä pysyy vakiona. Lisäksi jokainen järjestelmän prosessi on yhdessä tiloista s_0,...,s_4. Toisen paikkainvariantin i_2 avulla huomataan, että kaikille saavutettaville merkkauksille $M_n \in R(N, M_0)$ pätee yhtälö [60]:

$$M_0(s_2) + k \cdot M_0(s_4) + M_0(s_5) = M_n(s_2) + k \cdot M_n(s_4) + M_n(s_5) = k$$

Yhtälöstä nähdään, että lukevien prosessien lukumäärä, kirjoittavien prosessien lukumäärä kerrottuna k:lla, ja prosessien hallintapaikassa olevien merkkien lukumäärä yhteenlaskettuna on vakio kaikissa saavutettavissa merkkauksissa. Kyseisen vakion arvo on lisäksi k, mikä on alkumerkkauksessa paikan s_5 merkkien lukumäärä.

Edellä esitetyn yhtälön avulla voidaan havaita järjestelmän kannalta tärkeitä ominaisuuksia. Yhtälöstä nähdään, että paikassa s_4 voi olla vain yksi merkki, eli vain yksi prosessi voi olla samaan aikaan kirjoittamassa puskuriin. Mikäli yksi prosessi on kirjoittamassa, ei yksikään prosessi voi olla lukemassa tai siirtyä lukutilaan, sillä paikka s_5 ei tällöin sisällä yhtään merkkiä. Kun järjestelmän yksikään prosessi ei ole kirjoittamassa, jakautuu k kappaletta merkkejä paikkoihin s_2 ja s_5. Paikan s_2 merkkien lukumäärä kuvaa kyseisellä hetkellä lukevia prosesseja ja paikassa s_5 olevat merkit järjestelmän vapaita lukupaikkoja.

Paikkainvariantin i_2 avulla voidaan analysoida mallinnetun järjestelmän rajoittuvuusominaisuuksia. Paikka s_4 on rajoitettu, ja sen yläraja on yksi, eli $\forall M_n \in R(N, M_0) : M_n(s_4) \leq 1$. Paikat s_2 ja s_5 ovat myös rajoitettuja, ja niille pätee $\forall M_n \in R(N, M_0) : M_n(s_2) \leq k \wedge M_n(s_5) \leq k$.

Paikkainvarianttien avulla voidaan todistaa myös verkon elävyysominaisuuksia. Reisig [60] esittää kuinka kyseisen Petri-verkon elävyys todistetaan. Todistus tehdään yksinkertaisesti tutkimalla, että kaikista verkon tiloista voidaan päästä suorittamaan kaikki verkon siirtymät. Mikäli $M(s_0) + M(s_2) + M(s_4) > 0$, on vähintään yksi siirtymistä t_0, t_2, t_3 tai t_5 mahdollinen. Mikäli $M(s_0) + M(s_2) + M(s_4) = 0$ nähdään paikkainvariantista i_1, että $M(s_1) + M(s_3) = n$, ja paikkainvariantista i_2, että

$M(s_5) = k$. Tällöin siirtymä t_1 tai t_4 on mahdollinen. Edellä esitetyistä yhtälöistä nähdään, että mikäli $M(s_0) = 0$ voidaan kyseiseen paikkaan saada merkkejä suorittamalla jokin seuraavista siirtymäsarjoista: t_2, t_5, t_1t_2 tai t_4t_5. Näin ollen paikkaan s_0 voidaan aina saada merkkejä, mistä seuraa, että siirtymät t_0 ja t_3 ovat eläviä. Tästä seuraa edelleen kaikkien muidenkin siirtymien elävyys, joten koko Petri-verkko on elävä.

Paikkainvariantti i_2 antaa esimerkin invariantista, jonka paikat ovat painotettuja. Kyseisessä paikkainvariantissa paikan s_4 paino on k. Tässä tapauksessa paikan paino on nähtävissä yksinkertaisesti Petri-verkosta, sillä kyseiseen paikkaan tulee vain yksi kaari, ja tämän paino on k. Paikkainvariantin paikan paino voi olla yhdestä poikkeava muussakin tapauksessa. Tällöin paikkaan tulee tai sieltä lähtee kaari useampaan eri siirtymään.

Vektori $\vec{y} = (1, 1, 1, 0, 0, 0)^T$ on esimerkki kuvassa 3.7 esitetyn Petri-verkon siirtymäinvariantista. Vektori $\vec{y}$ vastaa lähtömerkkauksesta M_0 suoritettavaa tapahtumasarjaa $M_0[t_0\rangle M_1[t_1\rangle M_2[t_2\rangle M_0$. On siis olemassa sarja siirtymiä lähtömerkkauksessa, jotka suorittamalla päädytään takaisin kyseiseen merkkaukseen. Itse asiassa lähtömerkkaukseen päästään takaisin $\forall M_n \in R(N, M_0)$, mistä seuraa lähtömerkkauksen kotitilaominaisuus. Tutkimalla esitettyä Petri-verkkoa tarkemmin siirtymäinvarianttien avulla, huomattaisiin, että verkon jokainen tila on kotitila. Tämän todistaminen jätetään tekemättä, sillä osittain todistus tehtiin jo edellä paikkainvarianttien avulla.

4 Väritetyt Petri-verkot

Nykyään useimmin käytetyt Petri-verkot ovat korkean luokan Petri-verkkoja. Näistä suosituimmat ovat väritetyt Petri-verkot ja predikaatti-tapahtumaverkot niiden kohtuullisen laajan työkalutarjonnan sekä kohdealueriippumattomuuden ansiosta. Jensenin mukaan [36] nämä kaksi korkean luokan Petri-verkkoa ovat ennemminkin saman mallinnuskielen kaksi eri versiota kuin täysin erillisiä mallinnuskieliä.

Luku aloitetaan esittelemällä väritettyjen Petri-verkkojen sovellusalueita ja käyttöä ohjelmistokehityksessä. Seuraavaksi määritellään väritetyt Petri-verkot sekä niiden suorittamiseen liittyvät säännöt. Lisäksi esitetään väritettyjen Petri-verkkojen yhdistäminen hierarkkisiksi malleiksi. Lopuksi käydään läpi väritetyistä Petri-verkoista analysoitavia ominaisuuksia sekä näitä varten kehitettyjä analysointimenetelmiä.

4.1 Väritettyjen Petri-verkkojen sovellusalueet sekä käyttäminen osana ohjelmistokehitystä

Väritettyjä Petri-verkkoja on käytetty lukuisissa teollisissa ja akateemisissa projekteissa [16]. Yhtäaikaiset ohjelmat, protokollat sekä hajautetut järjestelmät ovat olleet koko Petri-verkkojen kehityksen ajan olennaisimpia sovelluskohteita. Varsinkin protokollien ja tietoverkkojen sekä hajautettujen järjestelmien mallintamisesta väritetyillä Petri-verkoilla on julkaistu monia tieteellisiä artikkeleita.

Artikkelissa [1] esitetään kuinka väritetyillä Petri-verkoilla on mahdollista mallintaa samanaikaisesti yhtäaikaisen ohjelman sekä oikeellisuus että tehokkuus. Monilla muilla menetelmillä voidaan yleensä keskittyä vain toiseen näistä. Artikkelissa [20] esitetään kuinka väritettyjä Petri-verkkoja voidaan käyttää myös protokollan verifiointiin. Samassa artikkelissa kuvataan väritetyn Petri-verkon tila-avaruuden avulla tehtyä analysointia. Tätä esitetään tarkemmin luvussa 4.4.2.

Yhtäaikaisten ohjelmien, protokollien ja hajautettujen järjestelmien mallintamisen lisäksi väritettyjä Petri-verkkoja on käytetty erilaisten ohjelmistojen sekä ohjausjärjestelmien mallintamiseen ja analysoimiseen. Kuten muitakin formaaleja menetelmiä, myös väritettyjä Petri-verkkoja sovelletaan usein kriittisten järjestelmien

mallintamisessa ja analysoimisessa.

1990-luvulta lähtien on monessa projektissa käytetty perinteisten väritettyjen Petri-verkkojen sijaan hierarkkisia väritettyjä Petri-verkkoja. Artikkelissa [63] kuvataan kuinka hierarkkisia väritettyjä Petri-verkkoja voidaan käyttää mallintamaan suodatinsirun toimintaa. Artikkeli esittää sirun mallintamisen yleiseltä tasolta aina rekisteritasolle asti. Hierarkkiset väritetyt Petri-verkot tarjoavatkin mahdollisuuden mallintaa samaan malliin havainnollisesti suunnitelma sekä yleisellä että yksityiskohtaisella tasolla. Tämä mahdollistaa suunnitelman havainnollisen tutkimisen eri abstraktiotasoilta sekä analysoinnin työkalun avulla. Hierarkkisia väritettyjä Petri-verkkoja kuvataan tarkemmin luvussa 4.3.

Korkean luokan Petri-verkkojen kehittäminen on lisännyt Petri-verkkojen soveltamista osana ohjelmistonkehitysprosessia. Reisig esittää artikkelissaan [61] kuinka Petri-verkkoja voidaan käyttää koko ohjelmiston ainoana suunnittelumenetelmänä. Tällöin tarkoituksena on esittää ohjelmiston käyttötapaukset Petri-verkkojen avulla, ja suunnittelun edistyessä koostaa näistä hierarkkisia malleja käsittäen koko ohjelmiston suunnitelman.

Laajojen ohjelmistojen suunnittelussa Petri-verkkoja ei kuitenkaan suositella ainoaksi suunnittelumenetelmäksi. Denaro ja Pezzè tutkivat artikkelissaan [13] ohjelmistotekniikan ja Petri-verkkojen yhteensopivuutta. Heidän mukaansa yksikään mallinnusmenetelmä ei yksinään riitä kuvaamaan ohjelmistoa sekä analysoinnin että suunnittelun kannalta siten, että tällä menetelmällä voitaisiin tuottaa ohjelmistosta selkeät näkymät eri tasoisille ihmisille, kuten asiakkaille ja suunnittelijoille. Tästä syystä myöskään väritettyjen Petri-verkkojen tarjoama näkökulma ohjelmistoon ei ole ainoana riittävä. Väritettyjen Petri-verkkojen kehittäjä Jensen toteaa artikkelissaan [32], ettei väritettyjä Petri-verkkoja ole kehitettykään ylivertaiseksi mallinnusmenetelmäksi. Kuitenkin monissa tapauksissa väritettyjen Petri-verkkojen käyttäminen on erittäin suositeltavaa.

UML (Unified Modeling Language) [51] on ollut viimeisen kymmenen vuoden ajan ohjelmistokehityksen ylivoimaisesti suosituin mallinusmenetelmä. UML sisältää kuitenkin puutteita ohjelmiston dynaamisten ominaisuuksien mallintamisessa ja analysoinnissa. Syynä tähän on se, että UML on pohjimmiltaan staattinen mallinnusmenetelmä. UML sisältää myös dynaamisia ominaisuuksia varten kehitettyjä kaavioita, mutta niiden suorittamiseen ei ole kehitetty formaalia semantiikkaa. Formaalin semantiikan puuttuminen estää formaalien analysointimenetelmien käytön UML-kaavioille. UML-kaavioita voidaan kuitenkin suorittaa tätä varten kehitetyil-

lä työkaluilla. Tällöin suorituksen semantiikka ja analysoitavat ominaisuudet ovat täysin työkalusta riippuvia [37].

UML:n staattisuudesta johtuen viime aikoina on tehty useita tutkimuksia ja käytännön sovellutuksia väritettyjen Petri-verkkojen käyttämisestä UML-kaavioiden dynaamiseen suorittamiseen ja analysointiin. Jørgensen esittää artikkelissaan [37], käytännön projektin antaman kokemuksen avulla, kuinka väritettyjä Petri-verkkoja voidaan käyttää yhdessä UML:n kanssa siten, että molempien mallinnusmenetelmien parhaat puolet saadaan käyttöön. Tällöin järjestelmän staattiset ominaisuudet voidaan mallintaa UML-kaavioilla ja dynaamiset ominaisuudet väritetyillä Petri-verkoilla.

Pinci ja Shapiro esittävät käytännön toteutuksen [57] avulla uudenlaisen ohjelmiston kehitysmenetelmän. Kehitysmenetelmän ideana on mallintaa järjestelmän vaatimukset ja niiden analysointi SADT-kaavioilla [44]. Analysoinnin jälkeen kaaviot muunnetaan automaattisesti väritetyiksi Petri-verkoiksi, joilla analyysivaiheen suunnitelmat verifioidaan ja järjestelmän suunnitteleminen toteutetaan. Edelleen väritetyistä Petri-verkoista generoidaan automaattisesti SML-kielistä ohjelmakoodia, jota muokkaamalla voidaan toteuttaa valmis järjestelmä.

Artikkelissa [57] esitetään, että analyysivaiheessa sekä toteutuksessa voidaan käyttää myös muita suunnittelumenetelmiä tai ohjelmointikieliä kuin SADT ja SML, esimerkiksi UML ja Java. Väritettyjen Petri-verkkojen muuntaminen ohjelmointikielelle ei kuitenkaan ole yleisessä tapauksessa yksinkertaista tai edes mahdollista ilman, että verkon elementeille määriteltäisiin tarkasti vastaavuus ohjelmointikielessä. Sen sijaan UML-kaavioiden muuntamiseen ohjelmointikielelle on olemassa valmiita työkaluja. Näin ollen käyttämällä suunnittelussa sekä UML-kieltä että väritettyjä Petri-verkkoja, voitaisiin ohjelmakoodin generointi toteuttaa yksinkertaisemmin.

4.2 Väritettyjen Petri-verkkojen peruskäsitteet

Luvussa määritellään väritetyt Petri-verkot sekä niiden suorittamiseen liittyvät säännöt. Lopuksi esitetään käytännön esimerkkinä yksinkertaisen tiedonsiirtoprotokollan määritteleminen ja suorittaminen väritettynä Petri-verkkona.

4.2.1 Väritettyjen Petri-verkkojen määrittely

Väritetyissä Petri-verkoissa merkit eivät kuvaa arvoa 1, vaan ne ovat monijoukkoja.

Määritelmä 4.1 *Monijoukko* $m : S \to \mathbb{N}$ *on funktio epätyhjästä ja äärellisestä joukosta* S *luonnollisiin lukuihin.*

Funktion $m(s) \in \mathbb{N}$ arvo kuvaa alkion $s \in S$ lukumäärää monijoukossa m [30]. Monijoukko esitetään yleisesti summana $\sum_{s \in S} m(s)'s$, jättäen summasta pois alkiot, joilla funktio m saa arvon 0. Esimerkiksi monijoukko $m(s) \mid s \in \{a, b, c\}$ määriteltynä $m(a) = 1$, $m(b) = 0$ ja $m(c) = 3$, esitetään $m = 1'a + 3'c$ [30]. Merkinnällä MS_{MS} tarkoitetaan kaikkia joukon MS monijoukkoja.

Esitellään vielä muutama merkintätapa ennen väritettyjen Petri-verkkojen määrittelemistä:

- $Type(v)$ kuvaa muuttujan v tyyppiä.
- $Type(expr)$ kuvaa lausekkeen $expr$ tyyppiä.
- $Var(expr)$ kuvaa lausekkeen $expr$ vapaiden muuttujien joukkoa.
- $Type(Var(expr))$ tarkoittaa joukkoa $\{Type(v) \mid v \in Var(expr)\}$.

Olkoon $Exprs$ kaikkien lausekkeiden joukko. Nyt voidaan formaalisti määritellä väritetty Petri-verkko.

Määritelmä 4.2 *Väritetty Petri-verkko (Coloured Petri Net) on yhdeksikkö* $(\Sigma, P, T, A, N, C, G, E, I)$ *[32], missä*

- Σ *on tyyppien joukko. Tyyppejä kutsutaan myös värien joukoiksi (colour sets).*
- P *on paikkojen joukko.*
- T *on siirtymien joukko.*
- A *on kaarien joukko.*
- N *on solmufunktio (node function)* $N : A \to (P \times T \cup T \times P)$.
- C *on värifunktio (colour function)* $C : P \to \Sigma$.

- *G on vahtifunktio (guard function)* $G : T \rightarrow Exprs$. *Vahtifunktiolle täytyy päteä* $\forall t \in T : [Type(G(t)) = \mathbb{B} \wedge Type(Var(G(t))) \subseteq \Sigma]$, *missä* $\mathbb{B}$ *on totuusarvojen joukko.*

- *E on kaarilausekkeen funktio (arc expression function)* $E : A \rightarrow Exprs$. *Kaarilausekkeen funktiolle täytyy päteä*
  $\forall a \in A : [Type(E(a)) = C(p)_{MS} \wedge Type(Var(E(a))) \subseteq \Sigma]$, *missä p on kaareen liittyvä paikka.*

- *I on alustusfunktio (initialisation function)* $I : P \rightarrow Exprs$ *siten, että*
  $\forall p \in P : [Type(I(p)) = C(p)_{MS}]$ *pätee.*

Lisäksi joukkojen, Σ, P, T *ja* A, *tulee olla äärellisiä, ja niille täytyy päteä* $P \cap T = P \cap A = T \cap A = \emptyset$ *ja* $\Sigma \neq \emptyset$.

Määritelmässä 4.2 esitetyt paikkojen P ja siirtymien T joukot ovat vastaavia kuin paikka-siirtymäverkossa. Lisäksi kaarien joukko A ja solmufunktio N, joka kuvaa kaaret lähtösolmusta päätesolmuun, vastaavat yhdessä paikka-siirtymäverkon kaarien joukkoa. Väritetyissä Petri-verkoissa kaaret kuvataan erillisen joukon ja funktion avulla sen takia, että kahden solmun välillä sallitaan useampi saman suuntainen kaari.

Väritetyn Petri-verkon arvojen, lausekkeiden ja funktioiden sallitut tyypit kuuluvat tyyppien joukkoon Σ. Jokaisella väritetyn Petri-verkon paikalla tulee olla tyyppi, jonka värifunktio C määrittelee. Funktio kuvaa minkä tyyppisiä arvoja kyseinen paikka voi sisältää.

Vahtifunktio G määrittelee totuusarvoisen lausekkeen siirtymän suorittamiselle. Vahtifunktion avulla voidaan tutkia millä muuttujien sidonnoilla siirtymä on mahdollinen. Esimerkiksi vahtifunktio siirtymälle $t_1 : G(t_1) = a > 1$ saa arvon tosi vain silloin kun kokonaislukumuuttujalle a sidotaan arvo, joka on suurempi kuin 1.

Kaarilausekkeen funktio E määrittelee jokaiselle kaarelle lausekkeen, jonka tyypin tulee olla kyseisen kaaren paikan värijoukon monijoukko. Suoritettaessa kaaren siirtymä, kaarilausekkeen funktiolla määrätään paikasta lähtevä tai paikkaan tuleva monijoukko. Se yleistää paikka-siirtymäverkon kaarien painofunktion väritetyille Petri-verkoille.

Alustusfunktio I määrittelee jokaiselle paikalle alkumerkkauksen, joka on kyseisen paikan värijoukon monijoukko. Tämä vastaa paikka-siirtymäverkon alkumerkkausta. Sen sijaan paikka-siirtymäverkon kapasiteettifunktiolle ei ole vastinetta väritetyissä Petri-verkoissa. Väritetyn Petri-verkon paikat voivat sisältää periaatteessa

äärettömän suuren monijoukon. Kapasiteettifunktion toimintaa voidaan haluttaessa mallintaa laskuripaikan avulla. Yleensä tällaisissa tapauksissa kannattaisi ennemmin muuttaa suunniteltua verkkoa siten, että kyseinen paikka olisi rajoitettu.

4.2.2 Väritettyjen Petri-verkkojen suorittaminen

Väritetyn Petri-verkon suorittaminen on samantapaista kuin muidenkin Petri-verkkojen. Väritetyn Petri-verkon suoritus kuitenkin poikkeaa luvussa 3.2.2 esitetystä paikka-siirtymäverkon suorittamisesta väritetyn Petri-verkon käyttämän tyyppijärjestelmän sekä vahtifunktion takia.

Ennen väritetyn Petri-verkon mahdollisten siirtymien määrittelyä, esitetään lausekkeen vapaiden muuttujien sidonnan määritelmä sekä esitystapa. Lausekkeen *expr* vapaiden muuttujien sidonta esitetään hakasuluissa

$$\langle v_1 = c_1, v_2 = c_2, \ldots, v_n = c_n \rangle, v_i \in Var(expr).$$

Esimerkiksi lausekkeen $2 * (x + 3y)$ vapaiden muuttujien x ja y sidonta saa aikaan monijoukon $(2 * (x + 3y))\langle x = b, y = d \rangle = 2'b + 6'd$.

Siirtymän sidonnan määrittelemiseksi esitellään kaksi uutta merkintätapaa kaikille siirtymille $t \in T$ [32]:

- $A(t) = \{a \in A \mid N(a) \in (P \times \{t\} \cup \{t\} \times P)\}$
- $Var(t) = \{v \mid v \in Var(G(t)) \vee (\exists a \in A(t) : v \in Var(E(a)))\}$

Funktio A palauttaa siirtymään t yhdistettyjen kaarien joukon. Tätä käytetään apuna määriteltäessä funktiota *Var*. Funktio *Var* palauttaa siirtymän t vahtifunktion, ja siirtymään yhdistettyjen kaarien kaarilausekkeiden funktioiden palauttamien lausekkeiden vapaiden muuttujien joukon. Funktion *Var* avulla saadaan siis kaikki siirtymän t suorittamiseen liittyvät vapaat muuttujat.

Edellä määriteltyjen funktioiden avulla voidaan määritellä väritetyn Petri-verkon siirtymän sidonta.

Määritelmä 4.3 *Siirtymän t sidonnan tyyppi on kaikkien siirtymään liittyvien vapaiden muuttujien tyyppien tulo [30]:*

$$BT(t) = Type(v_1) \times Type(v_2) \times \ldots \times Type(v_n), v_i \in Var(t)$$

Sidonnan tyypin avulla voidaan määritellä siirtymän t kaikkien mahdollisten sidontojen joukko [30].

Määritelmä 4.4 *Siirtymän t kaikkien mahdollisten sidontojen joukko on*

$$B(t) = \{(c_1, c_2, \ldots, c_n) \in BT(t) \mid G(t)\langle v_1 = c_1, v_2 = c_2, \ldots, v_n = c_n\rangle\}$$

Funktio B antaa siirtymän t kaikki sellaiset sidonnat, joilla vahtifunktio G palauttaa arvon tosi. Siirtymä t ei kuitenkaan ole välttämättä mahdollinen kaikilla mahdollisilla sidonnoilla. Myöhemmin määritellään mahdollinen siirtymä sidonnalla.

Määritellään vielä sidonta-alkion ja askeleen käsitteet sekä yleistetään väritetyille Petri-verkoille aikaisemmin paikka-siirtymäverkoille määritellyt merkin ja merkkauksen käsitteet. Jensen määrittelee edelliset käsitteet seuraavasti [30]:

- Väritetyn Petri-verkon *merkki (token element)* on pari (p, c), missä $p \in P$ ja $c \in C(p)$. Kaikkien merkkien joukkoa merkitään TE.
- *Sidonta-alkio (binding element)* on pari (t, b), missä $t \in T$ ja $b \in B(t)$. Kaikkien sidonta-alkioiden joukkoa merkitään BE.
- Väritetyn Petri-verkon merkkaus on monijoukko joukosta TE.
- *Askel (step)* on äärellinen ja epätyhjä monijoukko joukosta BE.

Väritetyn Petri-verkon merkin määritelmä kuvaa merkin paikan ja sen arvon, jonka tyypin tulee kuulua paikan värijoukkoon. Sidonta-alkiolla esitetään aikaisemmin määritelty siirtymä ja sen mahdollinen sidonta. Kaikkien sidonta-alkioiden joukosta voidaan muodostaa askel. Askeleella kuvataan yhden tai useamman siirtymän yhtäaikainen suorittaminen määritellyillä sidonnoilla. Mikäli askeleeseen kuuluu useampi kuin yksi siirtymä, sanotaan näiden siirtymien olevan yhtäaikaisia.

Nyt voidaan määritellä väritetyn Petri-verkon mahdollinen askel.

Määritelmä 4.5 *Askel Y merkkauksessa M on mahdollinen jos ja vain jos [30]*

$$\forall p \in P : \left[\sum_{(t,b) \in Y} E(p,t)\langle b \rangle \leq M(p) \right]$$

pätee.

Määritelmä 4.5 tarkoittaa sitä, että jokaisen paikan p sisältämien merkkien monijoukon tulee olla vähintään yhtä suuri kuin askeleen Y kaikkien sidonta-alkioiden ja paikan p kaarilausekkeiden monijoukkojen summa. Jokaisessa paikassa täytyy siis olla vähintään yhtä suuri monijoukko kun askeleen sieltä poistama monijoukko on. Määritelmä 4.5 vastaa paikka-siirtymäverkkojen määritelmää 3.3 sillä erotuksella, että väritettyjen Petri-verkkojen tapauksessa käsitellään monijoukkoja, eikä arvoa yksi kuvaavia merkkejä.

Askeleen suorittaminen saa aikaan merkkien poistamisen askeleeseen kuuluvien siirtymien esijoukkoon kuuluvista paikoista, ja merkkien lisäämisen siirtymien jälkijoukkoon kuuluviin paikkoihin. Merkkien määrät ja värit ovat riippuvaisia kyseisen paikan ja sidonta-alkion kaarilausekkeesta.

Määritelmä 4.6 *Merkkauksessa M_1 mahdollisen askeleen Y suorittaminen muuntaa merkkauksen M_1 uudeksi merkkaukseksi M_2 [30]:*

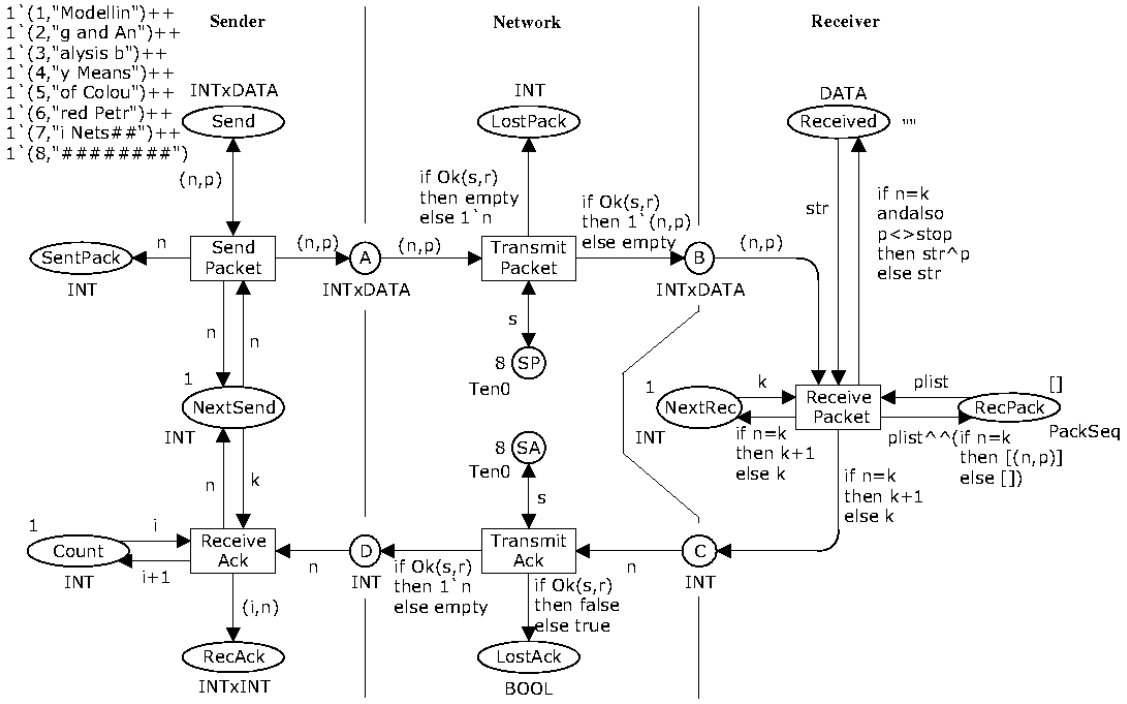
$$\forall p \in P : \left[M_2(p) = M_1(p) - \sum_{(t,b) \in Y} E(p,t)\langle b \rangle + \sum_{(t,b) \in Y} E(t,p)\langle b \rangle \right]$$

Väritetyille Petri-verkoille merkitään askeleen suorittamista ja sen muuntamaa merkkausta samaan tapaan kuin paikka-siirtymäverkoille $M_1[Y\rangle M_2$. Myös äärellinen $M_1[Y_1\rangle M_2 \ldots M_n[Y_n\rangle M_{n+1}$, ja ääretön tapahtumasarja $M_1[Y_1\rangle M_2[Y_2\rangle M_3 \ldots$ merkitään aikaisemmin esiteltyyn tapaan. Väritettyjen Petri-verkkojen tapauksessa kaikkia merkkauksesta M saavutettavia merkkauksia merkitään yleisesti $[M\rangle$.

4.2.3 Esimerkki: Yksinkertainen tiedonsiirtoprotokolla

Luvussa esitetään käytännön esimerkkinä yksinkertaisen tiedonsiirtoprokollan määrittäminen ja suorittaminen väritettynä Petri-verkkona. Kuvassa 4.1 on esitetty esimerkkinä käytetty väritetty Petri-verkko. Kyseinen väritetty Petri-verkko on hieman monipuolisempi versio Jensenin artikkelissa [33] esittämästä yksinkertaisesta protokollasta. Kuitenkin esitetty protokolla on erittäin yksinkertainen. Protokolla toimii periaatteella, että lähettäjä lähettää samaa pakettia niin kauan kunnes saa paketille kuittauksen, jonka jälkeen siirtyy lähettämään seuraavana jonossa olevaa pakettia.

Kuvassa 4.1 esitetty väritetty Petri-verkko sisältää 16 paikkaa ja viisi siirtymää. Väritetyn Petri-verkon vasen puoli kuvaa lähettäjää, keskialue verkkoa ja oikea puoli vastaanottajaa. Paikka-siirtymäverkkojen tapaan paikat on kuvattu ellipseinä tai



Kuva 4.1: Väritettyjen Petri-verkkojen esimerkkinä käytetty yksinkertaisen tiedonsiirtoprotokollan malli.

ympyröinä, ja siirtymät suorakulmioina. Luettavuuden takia kyseisen väritetyn Petri-verkon paikat ja siirtymät on nimetty.

Paikka-siirtymäverkoista poiketen kaarien kohdalle ei merkitä niiden painoa, vaan väritetyn Petri-verkon esittämisessä käytetään kaarilauseketta tyyppijärjestelmän takia. Samasta syystä väritetyn Petri-verkon merkkausta ei esitetä mustilla ympyröillä, vaan merkkaus on esitetty paikan vieressä monijoukkona. Väritettyjen Petri-verkkojen graafisessa esittämisessä voidaan myös näyttää paikkojen tyyppi. Paikan tyyppi esitetään merkkauksen tapaan paikan vieressä. Monimutkaisten kaarilausekkeiden, merkkausten ja paikkojen tyyppien esittäminen saattaa aiheuttaa väritetyn Petri-verkon ulkoasun sekavuuden. Tällä ei kuitenkaan ole käytännössä merkitystä väritettyjen Petri-verkkojen suorittamiseen, sillä verkon suorittamista graafisesti käytetään vain sitä rakennettaessa ja etsittäessä virheitä. Näissä tapauksissa työkalun vastuulle jää tarjota käyttäjälle selkeä kuva mallinnettavasta väritetystä Petri-verkosta.

Luvussa 4.2.1 esitettiin väritetyn Petri-verkon formaali määritelmä 4.2, jonka mukaan väritetty Petri-verkko on yhdeksikkö $(\Sigma, P, T, A, N, C, G, E, I)$. Määritellään

seuraavaksi kuvassa 4.1 esitetty väritetty Petri-verkko formaalissa muodossa.

Verkon tyyppien joukko $\Sigma = (INT, INT \times DATA, INT \times INT, BOOL, Ten0, Ten1, DATA, PackSeq)$, missä INT on määritelty kokonaisluvuksi, $DATA$ merkkijonoksi ja $BOOL$ totuusarvoksi. Tyyppi $INT \times DATA$ on tyyppien INT ja $DATA$ karteesinen tulo. Samaan tapaan määritellään tyyppi $INT \times INT$. Tyyppi $Ten0$ on kokonaisluku väliltä $0 \ldots 10$, $Ten1$ kokonaisluku väliltä $1 \ldots 10$, ja $PackSeq$ on määritelty tyypiltään listaksi tyypin $INT \times DATA$ arvoja.

Esimerkissä voidaan määritellä verkon paikkojen joukon P sisältävän kuvassa esitettyjen 16 paikan nimet. Samoin siirtymien joukko T määritellään sisältämään kuvassa esitettyjen viiden siirtymän nimet. Normaalisti verkon paikkojen ja siirtymien joukot sisältäisivät kyseisen paikan tai siirtymän yksilöivän tunnuksen, jolloin paikan tai siirtymän nimen muuttaminen ei vaikuttaisi millään lailla verkon määritelmään.

Verkon kaarien joukko A sisältää kuvassa esitetyt 30 kaarta. Solmufunktio N kuvaa kaaret muodossa (s, d), missä s on lähtö- ja d kohdesolmu. Kuvassa esitetyt kaksikärkiset nuolet kuvaavat kahta eri suuntaan, mutta samalla kaarilausekkeella määriteltyä kaarta solmujen välillä.

Värifunktio C määrittelee verkon paikoille tyypit. Esimerkin verkon värifunktio C määritellään seuravasti:

$$
\begin{array}{rclrcl}
C(SentPack) & = & INT & C(RecAck) & = & INT \times INT \\
C(Count) & = & INT & C(Send) & = & INT \times DATA \\
C(NextSend) & = & INT & C(B) & = & INT \times DATA \\
C(D) & = & INT & C(A) & = & INT \times DATA \\
C(NextRec) & = & INT & C(Received) & = & DATA \\
C(LostPack) & = & INT & C(LostAck) & = & BOOL \\
C(C) & = & INT & C(RecPack) & = & PackSeq \\
C(SP) & = & Ten0 & C(SA) & = & Ten0
\end{array}
$$

Esimerkin väritetyn Petri-verkon siirtymille ei ole erikseen määritelty vahtifunktioita. Tällöin vahtifunktio saa oletusarvoisesti kaikilla siirtymän sidonnoilla arvon tosi. Esimerkin tapauksessa vahtifunktiolla voitaisiin esimerkiksi estää yli 512 merkkiä pitkien pakettien lähetys asettamalla siirtymän *SendPacket* vahtifunktioksi `G(SendPacket)=length(p)<=512`, missä funktio `lenght` palauttaisi merkkijonon merkkien lukumäärän. Kyseisellä vahtifunktiolla yli 512 merkkiä sisältävät paketit eivät missään tapauksessa tulisi mahdollisten sidontojen joukkoon mukaan.

Esimerkin väritetyn Petri-verkon kaarilausekkeet on esitetty kaarien yhteydessä. Kaarilausekkeissa esitettyjen muuttujien tyypit ovat

$$\begin{aligned} Type(n) &= Type(k) = Type(i) = INT \\ Type(p) &= Type(str) = DATA \\ Type(s) &= Ten0 \\ Type(r) &= Ten1 \\ Type(plist) &= PackSeq \end{aligned}$$

Näiden lisäksi yhdessä kaarilausekkeessa esiintyy vakio `stop`, joka on määritelty merkkijonoksi `stop = '########'`. Kaarilausekkeissa käytetään myös funktiota `Ok`, joka on määritelty `Ok(s:Ten0,r:Ten1)=(r<=s)`. Funktio `Ok` ottaa siis parametrina kokonaisluvun väliltä $0 \ldots 10$ sekä toisen väliltä $1 \ldots 10$, ja palauttaa näiden vertailun totuusarvon. Esimerkin tapauksessa funktiolla `Ok` esitetään pakettien häviäminen verkossa.

Määritellään vielä esimerkin väritetyn Petri-verkon alustusfunktio I. Alustusfunktio I on määritelty seuraavasti:

$$\begin{aligned} I(Send) &= 1'(1,''Modellin'') + 1'(2,''g\ and\ An'') + 1'(3,''alysis\ b'')+ \\ &\quad 1'(4,''y\ Means'') + 1'(5,''of\ Colou'') + 1'(6,''red\ Petr'')+ \\ &\quad 1'(7,''i\ Nets\#\#'') + 1'(8,''\#\#\#\#\#\#\#\#'') \\ I(NextSend) &= 1'1 \qquad I(Count) = 1'1 \\ I(SP) &= 1'8 \qquad I(SA) = 1'8 \\ I(NextRec) &= 1'1 \qquad I(Received) = 1'('''') \\ I(RecPack) &= 1'[] \end{aligned}$$

Paikkaan *Send* alustetaan kahdeksan erillistä lähetettävää pakettia sisältäen paketin numeron ja tiedon. Paikkoihin *NextSend*, *Count*, *SP*, *SA* ja *NextRec* alustetaan kokonaislukuarvo. Näiden lisäksi paikka *Received* alustetaan tyhjällä merkkijonolla ja paikka *RecPack* tyhjällä listalla. Määrittelemättömät paikat sisältävät tyhjän monijoukon.

Kuvataan seuraavaksi kuvassa 4.1 esitetyn väritetyn Petri-verkon toimintaideaa. Lähettäjän puolella lähetykseen liittyviä paikkoja ovat *Send*, *SentPack* ja *NextSend*. Näistä *Send* sisältää lähetettävät paketit. Paikka *SentPack* pitää sisällään viimeisimmän lähetetyn paketin numeron, ja *NextSend* seuraavana lähetettävän paketin numeron. Paikka A toimii puskurina lähettäjän ja verkon välillä.

Siirtymä *SendPacket* voidaan suorittaa vain yhdellä mahdollisella sidonnalla. Tämä johtuu siitä, että paikka *NextSend* sisältää vain yhden kokonaislukuarvon, ja tämä arvo on muuttujan n ainoa mahdollinen sidonta. Näin ollen kaikkien siirtymään

SendPacket liitettyjen kaarien kaarilausekkeissa esiintyvä n voi saada vain paikassa *NextSend* olevan arvon. Samoissa kaarilausekkeissa esiintyvä muuttuja p sidotaan tällöin merkkijonolla, joka on numeron n sisältämässä paketissa.

Alkumerkkauksessa siirtymä *SendPacket* voidaan siis suorittaa vain ja ainoastaan sidonta-alkiolla $(SendPacket, \langle n{=}1, p{=}''Modellin''\rangle)$. Askeleen, joka sisältää kyseisen sidonta-alkion, suorittaminen poistaa merkin $(1,''Modellin'')$ paikasta *Send*, ja merkin 1 paikasta *NextSend*. Askeleen suorittaminen kuitenkin lisää samat merkit kyseisiin paikkoihin, sekä merkin 1 paikkaan *SentPack*, ja merkin $(1,''Modellin'')$ paikkaan A.

Verkon puolella oleva paikka *LostPack* sisältää kaikkien siirrossa hävitettyjen pakettien numeroiden monijoukon. Paikka B toimii puskurina verkon ja vastaanottajan välillä. Paikan SP sisältämää merkkiä käytetään verkon luotettavuuden määrittämisessä. Mikäli paikka A sisältää esimerkiksi merkin $(1,''Modellin'')$, voidaan siirtymä *TransmitPacket* suorittaa kymmenellä mahdollisella sidonnalla. Sidonta-alkioiden joukko sisältää sidonnat:

$$\begin{gathered}\langle n=1, p=''Modellin'', s=8, r=1\rangle\\ \langle n=1, p=''Modellin'', s=8, r=2\rangle\\ \vdots\\ \langle n=1, p=''Modellin'', s=8, r=10\rangle\end{gathered}$$

Muuttujien n ja p sidonta on tässä tapauksissa aina sama, sillä paikka A sisälsi vain yhden merkin. Mikäli paikka A olisi sisältänyt useamman merkin, voitaisiin myös muuttujat n ja p sitoa useilla eri arvoilla. Muuttuja s sidotaan aina paikan SP alustusfunktiossa määritetyllä arvolla, sillä kyseinen arvo ei voi muuttua verkkoa suoritettaessa. Esimerkin tapauksessa vain muuttuja r voidaan sitoa useilla eri arvoilla. Koska yksikään siirtymään *TransmitPacket* liittyvän paikan merkki ei sido arvoa muuttujalle r, voi se saada kaikki muuttujan tyypin mahdolliset arvot. Tässä tapauksessa muuttujan r tyyppi on $Ten1$, sillä funktion `Ok` toinen parametri tulee olla kyseistä tyyppiä. Näin ollen muuttuja r voidaan sitoa arvoilla väliltä $1 \ldots 10$.

Suoritettaessa siirtymä *TransmitPacket* jollakin sidonnalla, poistetaan paikasta A sidonnassa ollut merkki (n, p). Paikasta SP poistetaan ja siihen lisätään paikassa ollut merkki. Nyt sidonnasta riippuen paikkoihin *LostPack* ja B lisätään erilaisia merkkejä. Funktio `Ok` palauttaa arvon tosi mikäli muuttujalle r on sidottu pienempi tai yhtä suuri arvo kuin muuttujalle s. Funktion `Ok` palauttaessa arvon tosi lisätään

paikkaan B monijoukko $1'(n, p)$, missä n ja p sisältävät sidotut arvot. Muussa tapauksessa lisätään paikkaan *LostPack* monijoukko $1'n$. Paikassa *SP* olevalla arvolla voidaan näin ollen määritellä verkon yli onnistuneesti siirrettyjen pakettien todennäköisyys.

Esitetään seuraavaksi verkon loppujen siirtymien ja paikkojen toiminta pääpiirteittäin esittämättä mahdollisia sidonta-alkioita. Sidonta-alkiot muodostettaisiin muille siirtymille samaan tapaan kuin edellä on määritelty siirtymille *TransmitPacket* ja *SendPacket*.

Vastaanottajan puolella paikka *NextRec* sisältää seuraavana odotetun paketin numeron. Paikassa *Received* ylläpidetään vastaanotettujen pakettien konkatenoitua merkkijonoa, ja paikka *RecPack* sisältää kaikki vastaanotetut paketit listassa. Suoritettaessa siirtymä *ReceivePacket*, vastaanotetaan merkki paikasta B. Mikäli vastaanotetun merkin muuttuja n sisältää saman arvon kuin seuraavaksi odotetun paketin arvo, lisätään paikkaan *RecPack* vastaanotettu merkki, ja konkatenoidaan merkin sisältämä merkkijono paikan *Received* sisältämään merkkijonoon, mikäli kyseinen merkkijono ei ole lähetyksen lopettamista kuvaava merkkijonovakio *stop*. Lisäksi lähetetään seuraavana odotetun paketin numero vastaanottajalle siirtämällä numeron sisältämä merkki paikkaan C, ja kasvatetaan paikassa *NextRec* olevaa arvoa yhdellä. Mikäli vastaanotettu merkki ei sisällä odotettua paketin numeroa, paikkoihin *Received*, *RecPack* ja *NextRec* ei lisätä mitään uutta, ja odotetun paketin numero lähetetään vastauksena lähettäjälle.

Siirtymän *TransmitAck* suorittaminen toimii vastaavasti kuin aikaisemmin esitellyn siirtymän *TransmitPacket* suorittaminen. Tässä tapauksessa onnistuneesti siirrettyjen ja hävitettyjen pakettien lukumäärä lisätään paikan *LostAck* totuusarvoiseen monijoukkoon.

Lähettäjän päässä siirtymän *ReceiveAck* suorittaminen asettaa vastaanotetun merkin paikkaan *NextSend*, sekä kasvattaa paikan *Count* arvoa yhdellä. Kyseisen paikan tarkoituksena on ylläpitää lukumäärää vastaanotetuista Ack-viesteistä. Lisäksi paikkaan *RecAck* lisätään pari, joka sisältää vastaanotetun Ack-viestin järjestysnumeron sekä tämän arvon.

4.3 Hierarkkiset väritetyt Petri-verkot

Laajojen järjestelmien mallintaminen tavallisilla väritetyillä Petri-verkoilla aiheuttaa helposti liian monimutkaisten ja sekavien verkkojen muodostumisen, tai luke-

mattomien toisistaan erillisinä analysoitavien verkkojen mallintamisen. Tätä ongelmaa varten on kehitetty hierarkkiset väritetyt Petri-verkot. Hierarkkisten väritettyjen Petri-verkkojen avulla voidaan erilliset väritetyt Petri-verkot mallintaa yksinkertaisina, mutta siltikin käyttää niitä yhdessä muiden väritettyjen Petri-verkkojen kanssa koko järjestelmän analysoimiseen.

Ohjelmistotekniikan alkuajoista lähtien uudelleenkäyttöä on pidetty yhtenä tärkeimmistä ohjelmistokehitystä parantavista käytänteistä [42]. Hierarkkiset väritetyt Petri-verkot parantavat uudelleenkäytön mahdollisuutta myös Petri-verkkoihin perustuvissa kehitysmenetelmissä. Hierarkkisten väritettyjen Petri-verkkojen avulla kerran mallinnettua uudelleenkäytettävää väritettyä Petri-verkkoa voidaan käyttää helposti osana muiden järjestelmien suunnittelua, tai useaan kertaan saman järjestelmän suunnittelun yhteydessä.

Huber ym. esittävät artikkelissaan [23] viisi erilaista tapaa luoda väritettyjen Petri-verkkojen hierarkia. Hierarkkinen väritetty Petri-verkko voi sisältää useita näistä. Seuraavaksi määritellään näistä kaksi yleisimmin käytettyä, korvaavat siirtymät ja paikkojen fuusio, joita käytetään myös määriteltäessä hierarkkiset väritetyt Petri-verkot luvussa 4.3.3. Muut artikkelissa [23] esitetyt menetelmät ovat korvaavat paikat (substitution places), kutsuttavat siirtymät (invocation transitions), sekä siirtymien fuusio (fusion of transitions). Näitä ei käsitellä tarkemmin, sillä ne eivät ole olennaisia määriteltäessä hierarkkisia väritettyjä Petri-verkkoja.

4.3.1 Korvaavat siirtymät

Korvaavat siirtymät (substitution transitions) mahdollistavat hierarkkisen väritetyn Petri-verkon jakamisen yli- ja alisivuihin (superpage, subpage). Tarkoituksena on mahdollistaa hierarkkisen väritetyn Petri-verkon mallintaminen noudattaen joko top-down tai bottom-up lähestymistapaa. Top-down lähestymistavassa mallinnetaan verkkoa yleisimmältä tasolta lähtien kohti yksityiskohtaisempia kuvauksia. Bottom-up lähestymistapa on tälle vastakkainen, alkaen yksityiskohtaisimpien sivujen mallintamisesta, ja näitä yhdistellen, päätyen lopulta korkeimman tason kuvaukseen mallinnettavasta verkosta.

Hierarkkisen väritetyn Petri-verkon korvaavat siirtymät ovat eräänlaisia mustia laatikoita, joiden toiminnasta ei ole korvaavan siirtymän sisältävällä sivulla tarkkaa tietoa. Korvaavat siirtymät sisältävät sivun alisivun, jonka siirtymiä voidaan suorittaa sen sijaan, että itse korvaavaa siirtymää olisi mahdollista suorittaa. Korvaava siirtymä ja sen esi- ja jälkijoukko muodostavat siis erillisen alisivun.

Sivulla olevan korvaavan siirtymän esi- tai jälkijoukkoon kuuluvia paikkoja sanotaan *socket-solmuiksi*. Näistä esijoukkoon kuuluvia paikkoja kutsutaan *syöte (input)* socket-solmuiksi, ja jälkijoukkoon kuuluvia *tulos (output)* socket-solmuiksi. Vastaavasti alisivun paikkoja, joihin voidaan ylisivusta lisätä tai poistaa merkkejä, kutsutaan *syöte- ja tulosporttisolmuiksi (port node)* [23].

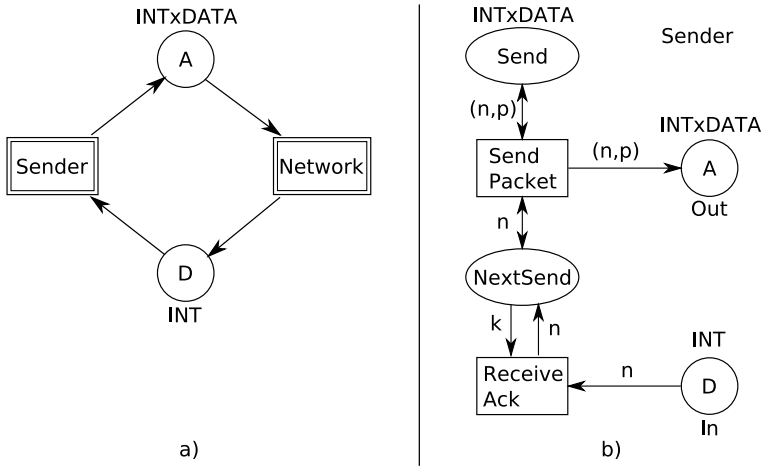
Hierarkkista väritettyä Petri-verkkoa mallinnettaessa määritellään portinmääräämisfunktio, joka yhdistää ylisivun socket-solmut alisivun porttisolmuihin. Socket- ja porttisolmut ovat keskenään suhteessa siten, että socket-solmun sisältämä merkki on myös portinmääräämisfunktion osoittamissa porttisolmuissa. Vastaavasti alisivun porttisolmuissa olevat merkit ovat vastaavissa ylisivun socket-solmuissa. Portinmääräämisfunktio määritellään tarkemmin määriteltäessä hierarkkiset väritetyt Petri-verkot luvussa 4.3.3.

Luvussa 4.2.3 esitettyä yksinkertaista tiedonsiirtoprotokollaa voidaan käyttää osana suuremman järjestelmän mallintamista. Tällöin protokollaa käytettäisiin korvaavan siirtymän alisivuna. Lähettäjän päässä protokollan paikka *Send* voitaisiin määritellä syöte porttisolmuksi, ja paikka *RecAck* tulos porttisolmuksi. Tällöin ylisivun kautta olisi mahdollisuus syöttää lisää paketteja protokollan paikkaan *Send* lähetettäväksi, ja edelleen vastaanottaa pakettien kuittaukset paikasta *RecAck*. Vastaanottajan puoli voitaisiin mallintaa samaan tapaan. Näin korvaavat siirtymät mahdollistavat mallinnettujen väritettyjen Petri-verkkojen uudelleenkäytön.

Kuvassa 4.2 esitetään osa luvussa 4.2.3 kuvatun protokollan yksinkertaistetusta hierarkkisesta versiosta. Kuvassa on esitetty pääsivu sisältäen kaksi korvaavaa siirtymää, sekä toisen korvaavan siirtymän sisältämä alisivu.

Kuva 4.2 a) esittää tiedonsiirtoprotokollaa yleisellä tasolla. Yleisestä kuvauksesta nähdään helposti mitä kyseisen hierarkkisen väritetyn Petri-verkon on tarkoitus mallintaa. Kuva sisältää kaksi korvaavaa siirtymää, *Sender* ja *Network*, jotka on esitetty kuvassa kaksinkertaisella reunuksella verrattuna tavallisiin siirtymiin. Korvaavia siirtymiä ei voida suorittaa, kuten aikaisemmin esitettiin. Kuvassa olevat paikat A ja D ovat molempien korvaavien siirtymien socket-solmuja. Paikka D on korvaavan siirtymän *Sender* syöte socket-solmu, ja paikka A tulos socket-solmu.

Kuva 4.2 b) esittää korvaavan siirtymän *Sender* sisältämää alisivua. Kyseinen alisivu kuvaa korvaavan siirtymän tarkempaa toimintaa. Alisivun paikka D on syöteporttisolmu, ja paikka A on tulosporttisolmu. Esimerkin tapauksessa portinmääräämisfunktio on määritelty yhdistämään pääsivun socket-solmu D alisivun porttisolmuun D. Vastaavaan tapaan on määritelty myös socket- ja porttisolmu A.



Kuva 4.2: Yksinkertaisen protokollan mallintaminen käyttäen korvaavia siirtymiä. a) Osa protokollan pääsivusta sisältäen korvaavat siirtymät *Sender* ja *Network*. b) Korvaavan siirtymän *Sender* sisältämä alisivu.

Välttämättä portinmääräämisfunktion ei tarvitse olla bijektio, vaan se voi yhdistää yhden socket-solmun useaan porttisolmuun tai toisinpäin. Korvaavien siirtymien määritelmä mahdollistaa myös socket- ja porttisolmut, joita ei ole yhdistetty yhteenkään portti- tai socket-solmuun. Lisäksi porttisolmut voivat olla myös syöte ja tulos -solmuja, minkä avulla mahdollistetaan verkon parempi uudelleenkäytettävyys [23].

Kuvassa 4.2 a) esitetty hierarkkinen väritetty Petri-verkko on mahdollista esittää ei-hierarkkisena väritettynä Petri-verkkona. Tällöin korvaava siirtymä *Sender* sekä siihen liitetyt kaaret korvataan kuvassa 4.2 b) esitetyllä ei-hierarkkisella väritetyllä Petri-verkolla. Alisivu asetetaan pääsivulle siten, että vastaavat socket- ja porttisolmut yhdistetään. Samaan tapaan toimitaan kaikille hierarkkisen väritetyn Petri-verkon korvaaville siirtymille.

4.3.2 Paikkojen fuusio

Paikkojen fuusio (fusion of places) tarkoittaa joukkoa paikkoja, joilla on aina keskenään sama merkkaus. Määritellään seuraavaksi fuusiojoukot.

Määritelmä 4.7 *Olkoon I kaikkien fuusiojoukkojen indeksien joukko. Kaikille fuusiojoukoille F_i pätee*

$$\forall i \in I : [\forall p_1, p_2 \in F_i, \forall M \in [M_0\rangle : M(p_1) = M(p_2)] .$$

Lisäksi jokainen paikka voi kuulua vain yhteen fuusiojoukkoon $\forall p_1, p_2 \in \cup_{F_i} : p_1 \neq p_2$.

Mikäli fuusiojoukkoon kuuluvaan paikkaan tulee tai poistuu merkki, lisätään tai poistetaan merkki myös kaikista samaan fuusiojoukkoon kuuluvista paikoista [23]. Fuusiojoukkoja voi olla kolmea eri tyyppiä [30]:

- *Globaali fuusiojoukko (global fusion set)* voi sisältää paikkoja hierarkkisen verkon jokaisen sivun jokaisesta instanssista.
- *Sivufuusiojoukko (page fusion set)* voi sisältää paikkoja vain samasta sivusta, mutta sen useista eri instansseista.
- *Instanssifuusiojoukko (instance fusion set)* voi sisältää paikkoja vain saman sivun samasta instanssista.

Instanssifuusiojoukko on yksinkertaisin fuusiojoukko. Sitä voidaan käyttää periaatteessa vain yksittäisen sivun yksinkertaisempaan mallintamiseen. Globaali ja sivufuusiojoukko mahdollistavat uudelleenkäytettävien sivujen käyttämisen ilman, että niitä tarvitsisi muokata mallinnettavaan hierarkkiseen väritettyyn Petri-verkkoon sopiviksi. Ilman globaalia tai sivufuusiojoukkoa, täytyisi monissa tapauksissa lisätä erillisiä paikkoja ja kaaria uudelleenkäytettävien sivujen välille, ja näin ollen sivuja jouduttaisiin aina muokkaamaan käyttötarpeeseen sopiviksi.

Sivufuusiojoukkoon kuuluvat paikat sisältävät saman merkkauksen sivun jokaisessa instanssissa. Tämä mahdollistaa esimerkiksi sovelluksen eri prosessien käyttämien globaalien asetusten mallintamisen. Samaan globaaliin fuusiojoukkoon kuuluvilla paikoilla on sama merkkaus koko hierarkkisessa väritetyssä Petri-verkossa. Tämä mahdollistaa esimerkiksi hajautetun järjestelmän usean sovelluksen käyttämän tietokannan mallintamisen.

4.3.3 Hierarkkisten väritettyjen Petri-verkkojen määrittely

Olkoon I äärellinen indeksien joukko, ja I^* kaikkien äärellisten indeksien sarjojen joukko. Aikaisemmin esiteltyjen ja määriteltyjen käsitteiden avulla voimme seuraavaksi määritellä hierarkkiset väritetyt Petri-verkot.

Määritelmä 4.8 *Hierarkkinen väritetty Petri-verkko (Hierarchical Coloured Petri Net) on kahdeksikko* $HCPN = (S, SN, SA, PN, PA, FS, FT, PP)$ *[30], missä*

- $S = \{S_i \mid i \in I\}$ *on sivujen äärellinen joukko. Joukolle* S *pätee* $\forall S_i \in S : S_i = (\Sigma_i, P_i, T_i, A_i, N_i, C_i, G_i, E_i, IN_i)$. *Lisäksi kaikkien sivujen paikat, siirtymät ja kaaret ovat erillisiä* $\forall i, k \in I : [i \neq k \Rightarrow (P_i \cup T_i \cup A_i) \cap (P_k \cup T_k \cup A_k) = \emptyset]$.

- $SN \subseteq T$ *on korvaavien siirtymien joukko, missä* $T = \bigcup_{i \in I} T_i$.

- SA *on sivunmääräämisfunktio (page assignment function) korvaavista siirtymistä sivuihin* $SA : SN \rightarrow S$. *Yksikään sivu ei saa olla itsensä alisivu:*
  $\{i_0, i_1, \ldots, i_n \in I^* \mid n \in \mathbb{N}_+ \wedge i_0 = i_n \wedge \forall k \in 1..n : S_{i_k} \in SA(SN_{i_{k-1}})\} = \emptyset$

- $PN \subseteq P$ *on porttisolmujen joukko, missä* $P = \bigcup_{i \in I} P_i$.

- PA *on portinmääräämisfunktio (port assignment function) korvaavista siirtymistä parien joukkoon. Parien alkiot yhdistävät socket-solmut ja porttisolmut* $PA(x) \subseteq X(x) \times PN_{SA(x)}$, *missä*
  $X(x) = \{x' \in (P \cup T) \mid \exists a \in A : [N(a) = (x, x') \vee N(a) = (x', x)]\}$ *palauttaa solmuun* x *yhdistetyt solmut. Jokaisella tulosjoukon alkiolla on keskenään sama värijoukko sekä yhtäläiset alustuslausekkeet:*
  $\forall x \in SN, \forall (x_1, x_2) \in PA(x) : [C(x_1) = C(x_2) \wedge IN(x_1)\langle\rangle = IN(x_2)\langle\rangle]$.

- $FS = \{FS_r\}_{r \in R}$ *on fuusiojoukkojen äärellinen joukko.* FS *on paikkojen* P *ositus, eli* $\forall x_1, x_2 \in \bigcup_{r \in R} \quad FS_r : x_1 \neq x_2$ *pätee, missä* R *on osituksen komponenttien joukko. Samaan fuusiojoukkoon kuuluvilla paikoilla tulee olla sama värijoukko sekä yhtäläiset alustuslausekkeet:*
  $\forall r \in R, \forall x_1, x_2 \in FS_r : [C(x_1) = C(x_2) \wedge IN(x_1)\langle\rangle = IN(x_2)\langle\rangle]$.

- FT *on fuusiotyypin funktio (fusion type function) fuusiojoukkojen joukosta joukkoon* $\{globaali, sivu, instanssi\}$, $FT : FS \rightarrow \{globaali, sivu, instanssi\}$. *Sivu ja instanssi tyyppisten fuusiojoukkojen alkiot kuuluvat samalle sivulle:*
  $\forall r \in R : [FT(FS_r) \neq globaali \Rightarrow \exists i \in I : FS_r \subseteq P_i]$.

- $PP \in S_{MS}$ *on ensisijaisten sivujen (prime pages) monijoukko.*

Hierarkkinen väritetty Petri-verkko koostuu joukosta ei-hierarkkisia väritettyjä Petri-verkkoja eli sivuja. Nämä ei-hierarkkiset väritetyt Petri-verkot kuuluvat joukkoon S. Jokaisesta yksittäisestä sivusta voi olla useita ilmentymiä. Sivun jakaminen

alisivuihin tehdään korvaavilla siirtymillä, jotka kuuluvat joukkoon SN. Sivunmääräämisfunktio SA yhdistää sivuilla olevat korvaavat siirtymät sivujen alisivuihin. Sivu ei kuitenkaan saa olla itsensä alisivu, sillä tällöin hierarkkisesta verkosta tulisi ääretön, eikä vastaavan ei-hierarkkisen väritetyn Petri-verkon muodostaminen olisi mahdollista.

Yksittäisten sivujen ilmentymien lukumäärä voidaan määrittää ensisijaisten sivujen monijoukon PP ja sivunmääräämisfunktion avulla. Ensisijaisten sivujen monijoukko määrittää hierarkkisen väritetyn Petri-verkon pääsivuihin kuuluvien sivujen lukumäärät. Nämä pääsivut voivat sisältää alisivuja korvaavien siirtymien avulla. Aloittaen ensisijaisten sivujen monijoukosta, voidaan sivujen ilmentymien lukumäärät laskea tästä rekursiivisesti sivunmääräämisfunktion avulla. Sivun jokainen ilmentymä on oma kopionsa sivusta, ja ne sisältävät omat merkkauksensa. Tähän poikkeuksen tekee paikkojen fuusio.

Hierarkkisen väritetyn Petri-verkon porttisolmut kuuluvat joukkoon PN. Portinmääräämisfunktio PA yhdistää nämä porttisolmut ja niiden ylisivuilla olevat socket-solmut. Näiden avulla pystytään määrittämään hierarkkisen väritetyn Petri-verkon eri sivujen merkkaus.

Määritelmä sisältää myös hierarkkisen väritetyn Petri-verkon fuusiojoukot. Nämä kuuluvat joukkoon FS. Hierarkkisen verkon jokainen paikka voi kuulua enintään yhteen fuusiojoukkoon, mutta suurin osa paikoista ei yleensä kuulu yhteenkään. Fuusiojoukkojen tyypit määrittelee fuusiotyypin funktio FT.

4.4 Väritettyjen Petri-verkkojen analysointi

Väritettyjen Petri-verkkojen analysoimiseen on kehitetty useita erilaisia menetelmiä. Menetelmät perustuvat yleensä kahteen erilaiseen analysointitapaan, tila-avaruuden analysointiin tai invarianttipohjaiseen analysointiin. Tässä luvussa esitetään molemmat tavat.

Luku aloitetaan määrittelemällä järjestelmistä analysoitavat ominaisuudet väritetyille Petri-verkoille. Tämän jälkeen esitellään kuinka ominaisuuksia voidaan analysoida tila-avaruuden avulla, ja käydään läpi tila-avaruuden analysoimista tehostavia menetelmiä. Lopuksi esitellään tila-avaruudesta riippumattomien paikka- ja siirtymäinvarianttien käyttäminen väritettyjen Petri-verkkojen analysoinnissa.

4.4.1 Analysoitavien ominaisuuksien yleistäminen väritettyihin Petri-verkkoihin

Väritetyistä Petri-verkoista analysoidaan samoja ominaisuuksia kuin luvussa 3.3 on esitetty Petri-verkoille yleisesti. Analysoitavan järjestelmän kannalta ominaisuudet ovat samoja kaikille Petri-verkoille, mutta ne tulee osittain yleistää väritetyille Petri-verkoille. Näistä saavutettavuus- ja kotitilaominaisuudet määritellään samaan tapaan myös väritetyille Petri-verkoille. Sen sijaan elävyys-, rajoittuvuus- ja reiluusominaisuudet yleistetään väritettyihin Petri-verkkoihin sopiviksi.

Määriteltäessä väritetyn Petri-verkon elävyysominaisuuksia ei tutkita pelkästään siirtymän vaan sidonta-alkioiden elävyyttä.

Määritelmä 4.9 *Jensen määrittelee merkkaukselle M ja joukolle sidonta-alkioita $X \subseteq BE$ elävyysominaisuudet seuraavasti [32]:*

- *M on kuollut mikäli $\forall x \in BE : \neg M[x\rangle$.*
- *X on kuollut merkkauksessa M mikäli $\forall M' \in [M\rangle$, $\forall x \in X : \neg M'[x\rangle$.*
- *X on elävä mikäli $\forall M' \in [M_0\rangle$, $\exists M'' \in [M'\rangle$, $\exists x \in X : M''[x\rangle$.*

Merkkauksen M sanotaan olevan kuollut mikäli yksikään sidonta-alkio ei ole siinä mahdollinen. Sidonta-alkioiden joukon X sanotaan olevan kuollut merkkauksessa M mikäli yksikään joukon X sidonta-alkio ei tule mahdolliseksi missään merkkauksesta M saavutettavassa merkkauksessa M'. Sidonta-alkioiden joukko X on puolestaan elävä mikäli ei ole olemassa saavutettavaa merkkausta, missä X olisi kuollut.

Siirtymän t elävyyttä voidaan analysoida määrittelemällä funktio $BE(t) \subseteq BE$, joka palauttaa kaikkien sidonta-alkioiden joukosta sellaiset sidonta-alkiot, joissa siirtymä t on mukana. Siirtymän elävyys määritellään, kuten joukon $BE(t)$ elävyys.

Väritetyistä Petri-verkoista voidaan analysoida kahdenlaisia rajoittuvuusominaisuuksia, merkkien lukumäärän ja monijoukon rajoittuvuutta.

Määritelmä 4.10 *Jensen määrittelee paikalle p ylärajat käyttäen luonnollista lukua $n \in \mathbb{N}$ sekä monijoukkoa $m \in C(p)_{MS}$ [32]:*

- *n on paikan p merkkien yläraja, mikäli $\forall M \in [M_0\rangle : |M(p)| \leq n$.*
- *m on paikan p monijoukon yläraja, mikäli $\forall M \in [M_0\rangle : M(p) \leq m$.*

Paikan p merkkien yläraja määritellään samaan tapaan kuin tavallisille Petri-verkoille. Sen sijaan väritettyjen Petri-verkkojen tapauksessa voidaan puhua myös paikan p monijoukon ylärajasta. Paikan p monijoukon yläraja on sellainen monijoukko, joka on suurempi tai yhtä suuri kuin paikan p monijoukko kaikissa saavutettavissa tiloissa. Vastaavasti voitaisiin määritellä paikan p merkkien ja monijoukon alaraja.

Samaan tapaan kuin elävyysominaisuudet yleistettiin väritetyille Petri-verkoille, voidaan myös reiluusominaisuuksia tutkia siirtymien sijasta sidonta-alkioiden kannalta. Olkoon $X \subseteq BE$ joukko sidonta-alkioita ja σ ääretön tapahtumasarja. Tällöin voidaan määritellä $EN_{X,i}(\sigma)$ kuvaamaan merkkauksessa M_i mahdollisten joukon X sidonta-alkioiden lukumäärää. Lisäksi määritellään $OC_{X,i}(\sigma)$ kuvaamaan askeleessa Y_i suoritettavien joukon X sidonta-alkioiden lukumäärää. Nyt voidaan määritellä tapahtumasarjassa σ mahdollisten ja suoritettavien joukon X sidonta-alkioiden lukumäärät [32]:

$$EN_X(\sigma) = \sum_{i=1}^{\infty} EN_{X,i}(\sigma) \qquad OC_X(\sigma) = \sum_{i=1}^{\infty} OC_{X,i}(\sigma)$$

Määritellään edellä esitettyjä määritelmiä apuna käyttäen sidonta-alkioiden joukon reiluusominaisuudet.

Määritelmä 4.11 *Jensen määrittelee [32] sidonta-alkioiden joukon $X \subseteq BE$ reiluusominaisuudet äärettömässä tapahtumasarjassa σ seuraavasti:*

- *tasapuolinen, mikäli* $OC_X(\sigma) = \infty$.
- *reilu, mikäli* $(EN_X(\sigma) = \infty) \Rightarrow (OC_X(\sigma) = \infty)$.
- *oikeudenmukainen, mikäli*
  $\forall i \geq 1 : [EN_{X,i}(\sigma) \neq 0 \Rightarrow \exists k \geq i : [EN_{X,k}(\sigma) = 0 \vee OC_{X,k}(\sigma) \neq 0]]$.

Määritelmässä 4.11 esitettiin, että sidonta-alkioiden joukko X on tasapuolinen äärettömässä tapahtumasarjassa σ, mikäli joukkoon X kuuluvien sidonta-alkioiden suoritusten lukumäärä on ääretön. Tämä vastaa perinteisille Petri-verkoille esitettyä määritelmää 3.11 luvussa 3.3.5.

Vastaavasti sidonta-alkioiden joukon X reiluus ja oikeudenmukaisuus äärettömässä tapahtumasarjassa σ määritellään samaan tapaan kuin perinteisille Petri-verkoille. Sidonta-alkioiden joukko X määriteltiin reiluksi äärettömässä tapahtumasarjassa σ, mikäli äärettömästä määrästä mahdollisia joukon X sidonta-alkioita myös seuraa äärettömän monta sidonta-alkioiden suoritusta.

Lopuksi sidonta-alkioiden joukko X määriteltiin oikeudenmukaiseksi äärettömässä tapahtumasarjassa σ. Mikäli X sisältää tietystä merkkauksesta lähtien jatkuvasti mahdollisen sidonta-alkion, niin joukkoon X kuuluva sidonta-alkio myös suoritetaan äärettömän useasti. Määritelmä ei kuitenkaan takaa, että joukkoon X kuuluva sidonta-alkio suoritettaisiin, mikäli jossakin merkkauksessa joukko X sisältäisi mahdollisen sidonta-alkion.

Siirtymän t reiluusominaisuuksia voidaan tutkia samaan tapaan kuin siirtymien elävyysominaisuuksia. Tällöin tutkitaan sidonta-alkioiden joukkoa $BE(t) \subseteq BE$, missä joukko $BE(t)$ sisältää kaikki siirtymään t liittyvät sidonta-alkiot.

4.4.2 Tapahtumagraafi

Tapahtumagraafi (Occurrence Graph, Reachability Graph, State Space) on verkko, joka sisältää kaikki väritetyn Petri-verkon tilat ja mahdolliset tilasiirtymät. Väritettyjen Petri-verkkojen tapahtumagraafia vastaa perinteisissä Petri-verkoissa kattavuuspuu, joka on esitelty luvussa 3.4.1. Kattavuuspuusta poiketen tapahtumagraafi ei itsessään sisällä minkäänlaisia tekniikoita graafin koon supistamiseen. Tapahtumagraafin supistamista varten on kuitenkin kehitelty lukuisia menetelmiä. Näitä esitellään luvussa 4.4.3.

Määritelmä 4.12 *Suunnattu verkko (directed graph) on pari (V, A), missä V on solmujen joukko, ja $A \in V \times V$ suunnattujen kaarien joukko [6].*

Tämä perinteinen määritelmä mahdollistaa kahden solmun välille vain yhden samansuuntaisen kaaren. Jensen antaa [32] tästä poikkeavan määritelmän suunnatuille verkoille siten, että kahden solmun välillä voi olla useita samansuuntaisia kaaria.

Määritelmä 4.13 *Suunnattu verkko, joka voi sisältää useita kaaria solmujen välillä, on kolmikko (V, A, N), missä V on solmujen joukko ja A kaarien joukko, joille pätee $V \cap A = \emptyset$. Lisäksi $N : A \rightarrow V \times V$ on funktio kaarista solmujen karteesiseen tuloon [32].*

Määritelmän 4.13 avulla voidaan määritellä täysi tapahtumagraafi.

Määritelmä 4.14 *Täysi tapahtumagraafi (full occurence graph, O-graph) on suunnattu verkko (V, A, N), missä [32]*

- $V = [M_0\rangle$.

- $A = \{(M_1, b, M_2) \in V \times BE \times V \mid M_1[b\rangle M_2\}$.
- $\forall a = (M_1, b, M_2) \in A : N(a) = (M_1, M_2)$.

Täyden tapahtumagraafin solmujen joukko sisältää kaikki väritetyn Petri-verkon saavutettavissa olevat tilat. Kaarien joukko sisältää verkon kaikki mahdolliset tilasiirtymät. Suunnattu kaari kuvaa verkon tilasiirtymää, joka muuntaa kaaren lähtösolmun tilan päätesolmun tilaksi suoritettaessa kaaren sidonta-alkio.

Esitetään seuraavaksi yksinkertainen algoritmi, jonka avulla voidaan muodostaa väritetyn Petri-verkon täysi tapahtumagraafi. Määritellään tätä varten muutama merkintätapa. W on joukko, joka sisältää tapahtumagraafin toistaiseksi käsittelemättömät solmut. $Node(M)$ luo tapahtumagraafiin uuden solmun M ja lisää tämän joukkoon W, mikäli tätä ei ole aikaisemmin luotu. $Arc(M_1, b, M_2)$ luo tapahtumagraafiin uuden suunnatun kaaren lähtösolmusta M_1 päätesolmuun M_2. Lisäksi määritellään funktio

$$Next(M_1) = \{(b, M_2) \in BE \times M \mid M_1[b\rangle M_2\},$$

joka palauttaa joukon kaikista tilassa M_1 mahdollisista sidonta-alkioista sekä tiloista johon sidonta-alkion suorittaminen johtaa. Nyt voidaan esittää täyden tapahtumagraafin muodostava algoritmi [32]:

$W := \emptyset$
$Node(M_0)$
repeat
  valitse jokin solmu $M_1 \in W$
  for all $(b, M_2) \in Next(M_1)$ **do**
    $Node(M_2)$
    $Arc(M_1, b, M_2)$
  end for
  $W := W \setminus \{M_1\}$
until $W = \emptyset$

Edellä esitetyn algoritmin suoritus päättyy vain siinä tapauksessa, että väritetty Petri-verkko on rajoitettu. Muussa tapauksessa algoritmin suoritus jatkuu ikuisesti tapahtumaverkkoa laajentaen. Jensenin mukaan [32] tämä ei kuitenkaan ole suuri ongelma, sillä suurin osa käytännön järjestelmistä toteutetuista väritetyistä Petri-verkoista ovat rajoitettuja. Usein oikeista järjestelmistä mallinnetut väritetyt Petri-verkot ovat kuitenkin niin monimutkaisia, että ongelmaksi täyden tapahtumagraafin muodostamisessa tulee tila- ja aikavaativuus.

Mikäli täyttä tapahtumagraafia ei voida muodostaa sen äärettömyyden tai suuren tila- ja aikavaativuuden takia, voidaan osittaista tapahtumagraafia kuitenkin käyttää järjestelmän analysoimiseen. Tällöin järjestelmää voidaan analysoida etsimällä virheitä osittaisesta tapahtumagraafista. Osittaisen tapahtumagraafin avulla ei kuitenkaan voida todistaa järjestelmän ominaisuuksia.

Toinen yksinkertainen ratkaisu on muuttaa edellä esitettyä algoritmia siten, että tapahtumagraafia muodostettaisiin tietylle tasolle saakka, esimerkiksi kymmenen lähtösolmun alaista tasoa. Muodostetusta tapahtumagraafin osasta voidaan edelleen valita tietty solmu tarkempaa tutkimista varten, ja muodostaa tästä lähtien tapahtumagraafia jälleen tietyn määrän tasoja eteenpäin. Tämä menetelmä sisältää vastaavat ongelmat kuin muut osittaiseen tapahtumagraafiin perustuvat menetelmät, mutta näin analysoija voi itse määritellä tila-avaruuden mielenkiintoiset osat tarkempaa tutkimusta varten.

Monesti laajat väritetyt Petri-verkot on koostettu modulaarisesti, jolloin voidaan tutkia järjestelmän ominaisuuksia erillisten osien täysien tapahtumagraafien avulla. Tällöin voidaan saada erittäin tarkkaa tietoa järjestelmän yksittäisten osien ominaisuuksista, ja näistä johtaa edelleen koko järjestelmää kattavia ominaisuuksia. Tämäkään menetelmä ei ole täydellinen, sillä järjestelmän osat riippuvat yleensä toisistaan ainakin lähtömerkkauksen suhteen. Modulaarista tapahtumagraafin analysointia on esitetty artikkelissa [8].

Monet edellä esitetyt ongelmat tapahtumagraafin aika- ja tilavaativuuden suhteen voidaan ratkaista käyttämällä erilaisia tila-avaruuden supistamistekniikoita. Näitä esitellään tarkemmin luvussa 4.4.3.

Täydestä tapahtumagraafista voidaan analysoida kaikki luvuissa 4.4.1 ja 3.3 esitetyt ominaisuudet, sillä se sisältää kaikki väritetyn Petri-verkon tilat. Yhtäaikaisuus on ainoa asia mitä tapahtumagraafin avulla ei voida analysoida väritetystä Petri-verkosta. Tämä johtuu siitä, että tapahtumagraafin kaaret kuvaavat vain yhden sidonta-alkion suorittamista kerrallaan, sen sijaan, että kaikkia mahdollisia yhtäaikaisesti suoritettavia sidonta-alkioiden kombinaatioita varten olisi oma kaari.

Tapahtumagraafi on järjestelmän suunnittelijan kannalta erittäin helppo apuväline analysointiin, sillä analysointi voidaan toteuttaa täysin automaattisesti. Tapahtumagraafin huonona puolena on puolestaan se, että sillä tehty analysointi pätee vain kiinnitetyillä järjestelmän parametreilla. Muutettaessa parametreja, kuten alkumerkkausta, joudutaan analysointi tekemään uudestaan. Jensenin mukaan [32] tämä aiheuttaa ongelmia kuitenkin erittäin harvoin, sillä yleensä järjestelmien tulisi

toimia samojen sääntöjen mukaan riippumatta annetuista parametreista.

Tiettyjä väritetyn Petri-verkon ominaisuuksia voidaan analysoida yksinkertaisemmin muodostamalla väritetyn Petri-verkon vahvasti yhtenäiset komponentit.

Määritelmä 4.15 *Vahvasti yhtenäinen komponentti (strongly connected component) on suunnatun verkon aliverkko, jonka jokaisesta solmusta on polku kaikkiin muihin samaan vahvasti yhtenäiseen komponenttiin kuuluviin solmuihin [6].*

Määritellään seuraavaksi vahvasti yhtenäisten komponenttien muodostama graafi, eli SCC-graafi. Olkoon SCC joukko, joka sisältää kaikki suunnatun verkon vahvasti yhtenäiset komponentit. Lisäksi merkintä v^c tarkoittaa sitä vahvasti yhtenäistä komponenttia, johon solmu v kuuluu. Määritellään vielä funktiot $s : A \rightarrow V$ ja $d : A \rightarrow V$, joista s palauttaa kaaren lähtösolmun ja d päätesolmun. Edellä annettujen merkintöjen avulla Jensen määrittelee SCC-graafin [32].

Määritelmä 4.16 *Suunnattu verkko (V^*, A^*, N^*) on suunnatun verkon (V, A, N) SCC-graafi mikäli pätee:*

1. $V^* = SCC$.
2. $A^* = \{a \in A \mid s(a)^c \neq d(a)^c\}$.
3. $\forall a \in A^* : N^*(a) = (s(a)^c, d(a)^c)$.

Suunnatun verkon SCC-graafin solmujen joukko sisältää verkon vahvasti yhtenäiset komponentit. Suunnatun verkon SCC-graafin kaarien joukkoon kuuluvat ne kaaret, jotka ovat vahvasti yhtenäisten komponenttien välillä. Näiden lisäksi funktio N^* on määritelty siten, että se palauttaa parin, joka sisältää SCC-graafin lähtösolmun ja päätesolmun, jotka ovat vahvasti yhtenäisiä komponentteja.

Väritetyn Petri-verkon ominaisuuksia voidaan analysoida tapahtumagraafista yksinkertaisesti käymällä kaikki tapahtumagraafin solmut läpi. Tiettyjä ominaisuuksia voidaan analysoida myös SCC-graafin avulla. Sekä solmujen läpikäynti että SCC-graafin muodostaminen voi suurissa verkoissa olla aikavaativuudeltaan suuri operaatio. Kumpikaan näistä ei kuitenkaan vie enempää aikaa kuin tapahtumagraafin luonti. Kaikkien solmujen läpikäynti on aikavaativuudeltaan luokkaa $O(v)$, ja SCC-graafin muodostaminen luokkaa $O(v + a)$, missä v on tapahtumagraafin solmujen ja a kaarien lukumäärä.

Jensen esittää kirjassa [31] ja artikkelissa [32] intuitiivisia todistussääntöjä, joilla täyden tapahtumagraafin, ja tämän SCC-graafin, avulla voidaan todistaa järjestelmien ominaisuuksia. Todistussääntöjä esitetään ja selitetään myös Christensenin ja Petruccin artikkelissa [8]. Todistukset näille säännöille löytyvät kirjasta [31]. Seuraavaksi esitetään todistussääntöjä jokaiselle luvussa 4.4.1 esitetylle ominaisuudelle.

Saavutettavuus: Kaikille tiloille $M_1, M_2 \in [M_0\rangle$ pätevät seuraavat saavutettavuusominaisuuksien todistussäännöt [31]:

- $[M_0\rangle = V$.
- $M_2 \in [M_1\rangle \Leftrightarrow \exists M_1[\sigma_1\rangle \ldots [\sigma_n\rangle M_2$.
- $|SCC| = 1 \Rightarrow M_2 \in [M_1\rangle$

Ensimmäisen saavutettavuuden todistussäännön mukaan vain ja ainoastaan ne tilat ovat saavutettavissa lähtötilasta, jotka kuuluvat täyden tapahtumagraafin solmujen joukkoon. Tämä on selvää jo täyden tapahtumagraafin määritelmän mukaan. Toisen todistussäännön mukaan tila M_2 on saavutettavissa tilasta M_1 vain mikäli täyden tapahtumagraafin solmusta M_1 on olemassa äärellinen polku solmuun M_2. Lopuksi kaikki tilat ovat saavutettavissa kaikista muista tiloista mikäli SCC-graafin koko on yksi, eli se sisältää vain yhden solmun. Tämä todistussääntö voidaan todistaa suoraan SCC-graafin ja vahvasti yhtenäisten komponenttien määritelmien 4.16 ja 4.15 avulla.

Elävyys: Olkoon SCC-graafin päätesolmujen joukko SCC_T. Päätesolmut ovat sellaisia solmuja, joista ei lähde yhtään kaarta. SCC-graafin päätesolmujen joukkoa SCC_T voidaan käyttää esitettäessä elävyyden todistussääntöjä. Kaikille saavutettaville tiloille $M \in [M_0\rangle$, sidonta-alkioille $X \subseteq BE$, ja siirtymille $t \in T$ pätevät seuraavat elävyysominaisuuksien todistussäännöt [31]:

- M on kuollut $\Leftrightarrow \forall a \in A, \forall M' \in [M_0\rangle : N(a) \neq (M, M')$.
- X on kuollut merkkauksessa $M \Leftrightarrow \forall b \in X, \forall a \in A, \forall M' \in [M_0\rangle : a \neq (M, b, M')$.
- X on elävä $\Leftrightarrow \forall c \in SCC_T : BE(c) \cap X \neq \emptyset$.
- $|SCC| = 1 \wedge \forall b \in BE(t) : (t, b) \in A \Rightarrow t$ on täysin elävä.

Väritetyn Petri-verkon tila M on kuollut mikäli siitä ei lähde täydessä tapahtumagraafissa yhtään kaarta. Vastaavasti sidonta-alkioiden joukko X on kuollut merkkauksessa M, mikäli ei ole olemassa sellaista kaarta, joka lähtisi merkkauksesta M ja sisältäisi jonkin sidonta-alkion joukosta X.

SCC-graafin avulla voidaan todistaa, että sidonta-alkioiden joukko X on elävä. Sidonta-alkioiden joukko X on elävä, mikäli jokaisessa SCC-graafin päätesolmussa on olemassa kaari, joka sisältää jonkin joukon X sidonta-alkion. Siirtymän t sanotaan olevan täysin elävä, kun se on mahdollinen kaikilla elävän sidonta-alkioiden joukon X alkioilla. Siirtymä t on täysin elävä, mikäli SCC-graafi sisältää vain yhden solmun, ja kaikki sidonta-alkiot, jotka sisältävät siirtymän t, kuuluvat täyden tapahtumagraafin kaarien joukkoon.

Rajoittuvuus: Kaikille paikoille $p \in P$ pätevät seuraavat rajoittuvuusominaisuuksien todistussäännöt [31]:

- Paikan p paras merkkien lukumäärän yläraja $= max\{|M(p)| \mid M \in V\}$.
- Paikan p paras monijoukon yläraja $= \sum_{c \in C(p)} max\{M(p,c) \mid M \in V\}'c$.

Väritetyn Petri-verkon paikan p paras merkkien yläraja saadaan ottamalla merkkien suurin lukumäärä, joka kyseiselle paikalle on täyden tapahtumagraafin solmujen joukossa. Vastaavasti verkon paikan p paras monijoukon yläraja saadaan etsimällä täyden tapahtumagraafin solmujen joukosta jokaiselle paikan p värille suurin arvo, ja laskemalla nämä monijoukot yhteen. Paikan merkkien lukumäärän ja monijoukon yläraja voidaan muuntaa alarajaksi vaihtamalla max-funktio min-funktioksi.

Kotitila: Kaikille merkkauksille $M \in [M_0\rangle$ pätevät seuraavat kotitilaominaisuuksien todistussäännöt [31]:

- M on kotitila $\Leftrightarrow SCC_T = \{M^c\}$
- $|SCC| = 1 \Leftrightarrow M_0$ on kotitila.

Kotitilaominaisuuksia on helppoa tutkia SCC-graafin avulla. Merkkaus M on kotitila mikäli SCC-graafin päätesolmujen joukko SCC_T sisältää vain sen solmun, johon merkkaus M kuuluu. Mikäli SCC-graafi sisältää vain yhden solmun, on alkumerkkaus M_0 kotitila. Tämä on selvää SCC-graafin määrittelyn takia. Vastaavasti kaikki täyden tapahtumagraafin merkkaukset ovat kotitiloja mikäli SCC-graafi sisältää vain yhden solmun.

Reiluus: Reiluusominaisuudet määriteltiin luvun 4.4.1 määritelmässä 4.11 äärettömälle tapahtumasarjalle. Täysi tapahtumagraafi voi sisältää äärettömän tapahtumasarjan vain solmujen silmukoina. Äärettömässä tapahtumasarjassa ei siis saada koskaan päätyä täyden tapahtumagraafin päätesolmuun.

Määritelmä 4.17 *Yksinkertainen äärellinen sykli on sellainen polku verkossa, jonka lähtö- ja päätesolmut ovat samoja, ja joka ei sisällä mitään muuta solmua kahdesti.*

Olkoon SFC täyden tapahtumagraafin kaikkien yksinkertaisten äärellisten syklien joukko. Lisäksi merkitään $BE(sc)$ kaikkia niitä sidonta-alkioita, jotka suoritetaan yksinkertaisessa äärellisessä syklissä sc. Vastaavasti merkitään $BE(M)$ kuvaamaan kaikkia merkkauksessa M mahdollisia sidonta-alkioita.

Nyt voidaan esittää todistussäännöt reiluusominaisuuksille. Kaikille sidonta-alkioiden joukoille $X \subseteq BE$ pätevät seuraavat reiluusominaisuuksien todistussäännöt [31]:

- X on tasapuolinen $\Leftrightarrow \forall sc \in SFC : [BE(sc) \cap X \neq \emptyset]$.
- X on reilu $\Leftrightarrow \forall sc \in SFC : [BE(sc) \cap X \neq \emptyset] \vee \forall M \in sc : BE(M) \cap X = \emptyset$.
- X on oikeudenmukainen $\Leftrightarrow \forall sc \in SFC : [BE(sc) \cap X \neq \emptyset] \vee \exists M \in sc : BE(M) \cap X = \emptyset$.

Täydestä tapahtumagraafista voidaan todistaa, että sidonta-alkioiden joukko X on tasapuolinen, mikäli jokaisessa yksinkertaisessa äärellisessä syklissä suoritetaan vähintään yksi joukon X sidonta-alkio. Tällöin äärettömässä tapahtumasarjassa tulee välttämättä ääretön määrä joukon X sidonta-alkioiden suorituksia. Joukon X ollessa tasapuolinen, se on myös reilu ja oikeudenmukainen.

Sidonta-alkioiden joukko X on reilu myös siinä tapauksessa, että yksikään sen alkio ei tule mahdolliseksi yhdessäkään merkkauksessa, joka kuuluu täyden tapahtumagraafin yksinkertaisten äärellisten syklien joukkoon. Vastaavasti sidonta-alkioiden joukko X voi olla myös oikeudenmukainen, mikäli täyden tapahtumagraafin jokainen yksinkertainen äärellinen sykli sisältää merkkauksen, jossa yksikään joukon X alkio ei ole mahdollinen.

4.4.3 Tila-avaruuden analysoinnin tehostaminen

Todellisten järjestelmien tilojen määrä on yleensä niin suuri, että edellisessä luvussa 4.4.2 esitetyn täyden tapahtumagraafin muodostaminen ei ole mahdollista. On-

gelmaa kutsutaan *tilojen määrän räjähtämisen ongelmaksi (state explosion problem)*. Ongelma esiintyy kaikenlaisissa tila-avaruuteen perustuvissa analysointimenetelmissä [68]. Ongelma ei siis liity pelkästään väritettyjen Petri-verkkojen analysointiin. Osittain tästä syystä, ongelmaa ratkaisemaan on kehitetty lukuisia erilaisia menetelmiä, joista useat ovat yleistettävissä moniin erilaisiin tilallisiin analysointimenetelmiin.

Luvussa esitetään kolme yleisesti käytettyä menetelmää väritettyjen Petri-verkkojen tila-avaruuden analysoinnin tehostamiseen: Symmetriamenetelmä, Pyyhkäisyviivamenetelmä ja Stubborn-joukot. Näiden lisäksi on olemassa suuri määrä muita menetelmiä.

Menetelmiä voidaan luokitella niiden toimintatavan mukaan. Esiteltävät Symmetriamenetelmä ja Stubborn-joukot perustuvat osittaisen tila-avaruuden muodostamiseen. Samaan luokkaan kuuluu kattavuuspuu, jota voidaan soveltaa väritettyjen Petri-verkkojen yhteydessä, samaan tapaan kuin luvussa 3.4.1 on esitetty. Kattavuuspuu supistaa vain rajoittamattomia tila-avaruuksia, ja niidenkin yhteydessä menetetään paljon tietoa.

Toisena luokkana on tila-avaruuden analysoinnin tekeminen modulaarisesti [8, 67]. Menetelmissä jokaiselle mallin erilliselle moduulille luodaan oma tila-avaruus, ja nämä yhdistetään koko järjestelmän tila-avaruudeksi. Tämä on osoittautunut modulaaristen mallien, kuten hierarkkiset väritetyt Petri-verkot, tapauksessa erittäin tehokkaaksi menetelmäksi.

Edellä mainituista menetelmistä poikkeavat muunnoksiin perustuvat menetelmät. Näissä menetelmissä pyritään alkuperäistä verkkoa muuntamaan yksinkertaisempaan muotoon siten, että tutkittava ominaisuus säilyy verkossa. Muunnoksiin perustuvilla menetelmillä tutkitaan kerralla vain yhtä tai paria verkon ominaisuutta. Haddad esittää [22] menetelmässään muunnoksia väritetyille Petri-verkoille.

Viimeiseen luokkaan kuuluvat menetelmät, joiden tarkoituksena on tehostaa täyden tapahtumagraafin muodostamista. Esimerkiksi Pyyhkäisyviivamenetelmä ja hajautettu tapahtumagraafin muodostaminen [40] kuuluvat tähän luokkaan. Nämä menetelmät ovat erittäin hyödyllisiä kohtuullisen ajan vaativien tapahtumagraafien muodostamisessa, mutta ne eivät yksinään riitä esimerkiksi eksponentiaalisesti kasvavien tila-avaruuksien analysoimiseen. Valmari esittää artikkelissaan [68] useita tässä mainitsematta jääneitä menetelmiä yleisen tilallisen järjestelmän analysoimiseen.

Jokainen menetelmä sisältää omat vahvuudet ja heikkoudet, eikä vielä ole ke-

hitetty menetelmää, joka olisi kaikissa tapauksissa riittävän hyvä. Tällaista menetelmää tuskin koskaan onnistutaankaan kehittämään tila-avaruuksien ja järjestelmistä analysoitavien ominaisuuksien suuren eroavuuden vuoksi. Kuitenkin monissa tapauksissa voidaan soveltaa useaa eri menetelmää yhdessä tehostamaan tila-avaruuden analysointia.

Symmetria ja ekvivalenssiluokat: Monet järjestelmät sisältävät symmetriaa tilojen välillä, ja tätä voidaan hyödyntää analysoitaessa järjestelmän tila-avaruutta. Esimerkiksi tietynlaisessa resurssienvarausjärjestelmässä ei ole väliä mikä monesta samanlaisesta resurssista otetaan käyttöön. Tällöin ei tarvitse luoda tila-avaruuteen tilaa jokaista samanlaisen resurssin varausta kohti, vaan riittää luoda yksi tila, joka esittää minkä tahansa identtisen resurssin varausta.

Merkkausten ja sidonta-alkioiden symmetria määritellään symmetriafunktioiden avulla. Olkoon Φ kaikkien symmetriafunktioiden joukko. Tällöin funktion $\phi \in \Phi$ sanotaan olevan symmetria, joka kuvaa merkkauksen M symmetriamerkkaukseksi $\phi(M)$. Kahden merkkauksen, m_1 ja m_2, sanotaan olevan ekvivalentteja, mikäli $m_1 = \phi(m_2)$ [24]. Tällöin merkitään $m_1 \sim m_2$, ja merkkauksien m_1 sekä m_2 sanotaan kuuluvan samaan ekvivalenssiluokkaan.

Samaan tapaan määritellään symmetriafunktio sidonta-alkioille. Kaikille kaarilausekkeille ja sidonnoille $b \in B(t)$ pätee

$$E(a)\langle\phi(b)\rangle = \phi(E(a)\langle b\rangle).$$

Tämä tarkoittaa sitä, että symmetrisillä sidonnoilla on keskenään symmetriset vaikutukset verkon merkkaukseen. Tästä saadaan tila-avaruuden kannalta olennainen dynaaminen ominaisuus symmetrialle.

Määritelmä 4.18 *Kaikille merkkauksille $M', M'' \in [M_0\rangle$, kaikille sidonta-alkioille $b \in BE$, ja kaikille symmetrioille $\phi \in \Phi$ pätee [32]*

$$M'[b\rangle M'' \Leftrightarrow \phi(M')[\phi(b)\rangle\phi(M'').$$

Määritelmän 4.18 mukaan symmetrisillä merkkauksilla on symmetriset sidonta-alkiot, ja edelleen sidonta-alkio suorittamalla symmetriset tulosmerkkaukset.

Symmetrian avulla voidaan täyden tapahtumagraafin sijaan muodostaa symmetriaa käyttävä tapahtumagraafi (OS-graph, Occurence Graph with Symmetries). Tämä tapahtumagraafi sisältää solmun jokaista merkkausten ekvivalenssiluokkaa, ja kaaren jokaista sidonta-alkioiden ekvivalenssiluokkaa kohti. Symmetriaa käyttävän tapahtumagraafin avulla voidaan todistaa lähes kaikki ominaisuudet, kuten täyden tapahtumagraafin avulla [32].

Symmetriaa käyttävän tapahtumagraafin avulla voidaan saada aikaan merkittävää tila-avaruuden supistumista. Jensen esittää artikkelissa [32] kuinka mallinnetun hajautetun tietokannan tila-avaruuden solmujen lukumäärä saadaan supistumaan täyden tapahtumagraafin luokasta $O(n \cdot 3^n)$ luokkaan $O(n^2)$, missä n on erillisten tietokantojen lukumäärä. Näin suuri supistuminen saadaan aikaan vain erittäin paljon symmetriaa sisältävässä järjestelmässä. Symmetrian käyttö ei kuitenkaan pienennä täysin ilman symmetriaa olevan järjestelmän tila-avaruutta. Yleensä järjestelmät kuitenkin sisältävät ainakin jonkin verran symmetriaa.

Pyyhkäisyviivamenetelmä: *Pyyhkäisyviivamenetelmän (Sweep-Line Method)* tarkoituksena on tehostaa luvussa 4.4.2 esitettyä täyden tapahtumagraafin muodostamista ja analysointia. Pyyhkäisyviivamenetelmä analysoi järjestelmän ominaisuuksia täyden tapahtumagraafin muodostamisen aikana, mikä mahdollistaa tila- ja aikavaativuuden pienentämisen. Menetelmän perusversio esitettiin Christensenin ym. artikkelissa [7].

Perinteinen menetelmä täyden tapahtumagraafin muodostamiseen vaatii paljon muistia, sillä koko tapahtumagraafi säilytetään muistissa. Lisäksi tilojen suuresta määrästä johtuen, perinteisen menetelmän funktio $Node(M)$ vie paljon prosessointiaikaa, sillä kyseisessä funktiossa tarkistetaan, ettei luotua tilaa ole jo olemassa. Pyyhkäisyviivamenetelmän täyden tapahtumagraafin muodostamisen aikainen analysointi mahdollistaa luotujen tilojen poiston, mikä pienentää muistin tarvetta, ja nopeuttaa jo olemassa olevien tilojen tarkistamista.

Pyyhkäisyviivamenetelmässä tehtävä luotujen tilojen poisto perustuu *edistymisen mittaan (measure of progress)*. Tila-avaruuden tilojen edistymisen mitta on sellainen arvo, jonka avulla tiloja voidaan luokitella. Edistymisen mitan tulee olla sellainen, että tiloista ei päästä oman edistymisen mitan arvoa pienempiin tiloihin. Tilasta, jonka edistymisen mitan arvo on n, voidaan päästä vain tiloihin, joilla edistymisen mitan arvo on suurempi tai yhtä suuri kuin n.

Edistymisen mittana voidaan luvussa 4.2.3 esitetyssä protokollassa käyttää esimerkiksi seuraavana lähetettävän paketin sekvenssinumeroa. Tämä numero ei voi protokollan suorituksen aikana pienentyä, joten tiloihin, joissa sekvenssinumero on nykyistä pienempi, ei voida enää päästä. Joissakin mallinnusmenetelmissä, kuten ajastetut väritetyt Petri-verkot [39], edistymisen mitta on sisäänrakennettuna.

Pyyhkäisyviivamenetelmässä tehdään luoduille tiloille halutuin välein roskienkeruu. Roskienkeruussa poistetaan kaikki ne tilat, joilla on nykyistä edistymisen mittaa pienempi arvo. Näin voidaan pitää tila-avaruuden analysoinnin käyttämä

muistin määrä pienenä. Lisäksi on havaittu, että algoritmin vaatima ylimääräinen työ on pienempi kuin pienemmän tila-avaruuden käsittelystä saatava hyöty. Christensen ym. saavat artikkelissaan [7] esitetyille kolmelle esimerkkitapaukselle muistin kulutuksen alta 10 prosenttiin ja suoritusajan alta 30 prosenttiin verrattuna perinteiseen menetelmään. Nämä molemmat riippuvat paljon verkon rakenteesta, ja siitä kuinka usein roskienkeruu tehdään.

Aika- ja tilavaativuuden pienentämisen lisäksi pyyhkäisyviivamenetelmän aikana voidaan hakea verkon vahvasti yhtenäiset komponentit. Nämä saadaan roskienkeruun yhteydessä, sillä jokaisella samaan vahvasti yhtenäiseen komponenttiin kuuluvalla tilalla on sama edistymisen mitan arvo. Verkon vahvasti yhtenäisiä komponentteja voidaan käyttää monien erilaisten ominaisuuksien analysoimiseen, kuten luvussa 4.4.2 on esitetty.

Pyyhkäisyviivamenetelmää ei voida käyttää sellaisten järjestelmien analysoimiseen, joille ei voida määrittää edistymisen mittaa. Menetelmä ei myöskään toimi tehokkaasti järjestelmillä, joissa on vain vähän vahvasti yhtenäisiä komponentteja, sillä tällöin tila-avaruuden puhdistamista voidaan tehdä vain harvoin. Lisäksi menetelmän huonona puolena on se, että tietyissä erikoistapauksissa joudutaan poistettuja tiloja luomaan uudestaan tai kaikkia poistettavissa olevia tiloja ei voida poistaa. Näin joudutaan menettelemään esimerkiksi deadlock-tilaan johtaneen polun etsinnässä.

Stubborn-joukot: *Stubborn-joukkojen (Stubborn Sets)* avulla pyritään vähentämään yhtäaikaisuuden esiintymistä tila-avaruudessa, samaan tapaan kuin symmetrian avulla voidaan vähentää symmetriaa tila-avaruudessa. Stubborn-joukkojen avulla luodaan vain osittainen tila-avaruus, kuten muissakin tila-avaruuden supistamiseen perustuvissa menetelmissä. Tämä mahdollistaa huomattavasti pienemmän tila-avaruuden luomisen, mutta vaikeuttaa sen analysoimista.

Stubborn-joukkojen tarkoituksena on muodostaa tapahtumagraafi, jossa on mahdollisimman vähän yhtäaikaisia tapahtumasarjoja. Yhtäaikaiset tapahtumasarjat ovat sellaisia siirtymien sarjoja, joissa siirtymien suoritusjärjestyksellä ei ole merkitystä lopputuloksen kannalta. Esimerkiksi kolme toisistaan riippumatonta siirtymää voidaan suorittaa kuudessa eri järjestyksessä, vaikka kaikkien eri kombinaatioiden jälkeen päädytään samaan tapahtumagraafin tilaan. Täysi tapahtumagraafi sisältää nämä kaikki kuusi erilaista polkua, mutta stubborn-joukkojen avulla nämä voidaan esittää yhden polun avulla.

Stubborn-joukko $T_s \subseteq T$ on joukko siirtymiä käsiteltävässä merkkauksessa M.

Mikäli merkkauksessa M on mahdollisia siirtymiä, tulee myös stubborn-joukon T_s sisältää vähintään yksi mahdollinen siirtymä [66]. Stubborn-joukkoihin kuuluvat siirtymät riippuvat tapahtumagraafista analysoitavista ominaisuuksista. Tämä johtuu siitä, että stubborn-joukkojen avulla tapahtumagraafiin muodostetaan seuraavia tiloja vain stubborn-joukkoon kuuluvien mahdollisten siirtymien mukaan, sen sijaan, että muodostettaisiin seuraajatiloja kaikkien tilassa mahdollisten siirtymien mukaan.

Erilaisista analysoitavista ominaisuuksista johtuen stubborn-joukoista on olemassa useita versioita. Kaikille versioille pätee yleisesti kuitenkin seuraava määritelmä.

Määritelmä 4.19 *Olkoon M väritetyn Petri-verkon merkkaus, T_s stubborn-joukko merkkauksessa M, $n \geq 0$, $t \in T_s$, ja $t_1, t_2, \ldots, t_n \notin T_s$, tällöin pätee [41]*

1. *Mikäli $M[t_1\rangle M_1[t_2\rangle \ldots [t_n\rangle M_n[t\rangle M'_n$, on $M[t\rangle$.*
2. *Mikäli $M[t_1\rangle M_1[t_2\rangle \ldots [t_n\rangle M_n$ ja $M[t\rangle M'$, on olemassa $M'_1, M'_2, \ldots, M'_n$ siten, että $M'[t_1\rangle M'_1[t_2\rangle \ldots [t_n\rangle M'_n$, ja $M_n[t\rangle M'_n$.*

Määritelmän 4.19 ensimmäisen säännön mukaan merkkauksessa M mahdollinen stubborn-joukkoon kuuluva siirtymä pysyy mahdollisena, vaikka suoritettaisiin stubborn-joukkoon kuulumattomia siirtymiä. Toisen säännön mukaan ei ole väliä missä järjestyksessä stubborn-joukkoon kuuluva, ja siihen kuulumattomat siirtymät suoritetaan. Tätä voidaan havainnollistaa seuraavalla kaaviolla [68]:

$$\begin{array}{ccccccc} M & -t_1 \rightarrow & M_1 & - t_2 \rightarrow & \ldots & - t_n \rightarrow & M_n \\ | & & & & & & | \\ t & & & & & & t \\ \downarrow & & & & & & \downarrow \\ M' & -t_1 \rightarrow & M'_1 & - t_2 \rightarrow & \ldots & - t_n \rightarrow & M'_n \end{array}$$

Stubborn-joukkojen hyvänä puolena on niiden mahdollistama suuri tila-avaruuden supistaminen. Valmari esittää artikkelissaan [68] kuinka stubborn-joukoilla voidaan supistaa mallinnetun hajautetun tietokannan tilojen lukumäärä luokasta $O(n \cdot 3^n)$ luokkaan $O(n^2)$. Hajautetun tietokannan mallina käytettiin samaa kuin edellä esitettiin symmetriamenetelmän yhteydessä. Molemmilla menetelmillä päästään samaa kertaluokkaa olevaan supistumiseen. Stubborn-joukot ja symmetriamenetelmä eivät kuitenkaan ole toisensa poissulkevia menetelmiä, vaan niitä voidaan soveltaa yhdessä. Tällöin Valmari pääsee hajautetun tietokannan tilojen lukumäärässä luokkaan $O(n)$.

Stubborn-joukkojen ongelmana on se, että erilaisia analysoitavia ominaisuuksia varten joudutaan stubborn-joukot määrittelemään erikseen. Näin ollen yhdestä stubborn-joukkojen avulla muodostetusta tapahtumagraafista ei voida analysoida kaikkia järjestelmän ominaisuuksia. Lisäksi Kristensenin ja Valmarin mukaan [41] tietyissä tapauksissa väritettyä Petri-verkkoa tulee muuntaa vastaavaksi paikka-siirtymäverkoksi, jotta paras mahdollinen supistuminen saadaan aikaan. Valitettavasti tämä operaatio on työläs, ja siten hidastaa tapahtumagraafin muodostamista.

4.4.4 Väritettyjen Petri-verkkojen paikka- ja siirtymäinvariantit

Väritettyjen Petri-verkkojen paikka- ja siirtymäinvarianttien periaatteet ovat vastaavat kuin luvussa 3.4.3 esitettiin perinteisille Petri-verkoille. Järjestelmien analysoinnin kannalta tarkoituksena on muodostaa paikkojen merkkauksien avulla yhtälöitä, jotka pätevät kaikissa saavutettavissa merkkauksissa. Vastaavasti voidaan muodostaa tapahtumasarjoja, joita suorittamalla päädytään takaisin ennen tapahtumasarjan suorittamista olleeseen lähtömerkkaukseen.

Väritettyjen Petri-verkkojen paikka- ja siirtymäinvariantit poikkeavat perinteisten Petri-verkkojen vastaavista siinä, että väritettyjen Petri-verkkojen paikkojen tyypit voivat poiketa kokonaisluvuista. Tämä vaikuttaa myös paikka- ja siirtymäinvarianttien määrittelyyn. Määritellään seuraavaksi väritettyjen Petri-verkkojen paikkainvariantit.

Eri tyyppisissä paikoissa olevia merkkejä ei voida laskea suoraan yhteen, kuten perinteisten Petri-verkkojen paikkainvarianteissa tehdään. Väritettyjen Petri-verkkojen tapauksessa täytyy paikkojen merkkejä varten muodostaa funktioita, jotka palauttavat samantyyppisiä arvoja. Tällaista funktiota kutsutaan paikan *painoksi (weight)*.

Yhtälöitä muodostettaessa voidaan joutua käsittelemään vähennyslaskua. Tämä ei kuitenkaan ole kaikissa tapauksissa määritelty monijoukoille. Määritellään tätä varten painotetut joukot.

Määritelmä 4.20 *Painotettu joukko (weighted set) on funktio* $WS : S \rightarrow \mathbb{Z}$ *paikkojen joukosta kokonaislukujen joukkoon [32].*

Painotetun joukon määritelmä 4.20 poikkeaa monijoukon määritelmästä 4.1 vain siinä, että se voi sisältää myös negatiivisia arvoja. Näin painotettujen joukkojen vähennyslasku on kaikissa tapauksissa määritelty.

Olkoon A_{ws} kaikkien painotettujen joukkojen joukko tyypissä $A \subseteq \Sigma$. Paikkainvariantteja varten määritellään jokaiselle paikalle $p \in P$ paino $W_p : C(p) \rightarrow A_{ws}$,

joka on funktio paikan väristä halutun tyypin painotettuun joukkoon. Kun kaikille paikoille määritellään paino samantyyppiseksi, voidaan kaikkien paikkojen painot laskea yhteen. Näin saadaan merkkaukselle *M painotettu summa (weighted sum)* [32]

$$W(M) = \sum_{p \in P} W_p(M(p)).$$

Painotettu summa ei saa olla riippuvainen merkkauksesta, eli yhtälön

$$\forall M \in [M_0\rangle : W(M) = W(M_0)$$

täytyy päteä. Edellisen yhtälön laskemisen helpottamiseksi määritellään paikkavuot.

Määritelmä 4.21 *Painojen joukko $W = \{W_p\}_{p \in P}$ on paikkavuo (place flow) mikäli pätee [32]*

$$\forall (t, b) \in BE : \sum_{p \in P} W_p(E(p, t)\langle b\rangle) = \sum_{p \in P} W_p(E(t, p)\langle b\rangle).$$

Paikkavuon tarkoituksena on tutkia, että jokaisen sidonta-alkion suoritus sekä poistaa että lisää verkkoon yhtä monta painotettua merkkiä. Tällöin kaikkien mahdollisten merkkausten painotettu summa pysyy samana.

Määritelmä 4.22 *Paikkavuo W määrittelee paikkainvariantin mikäli [45, 32]*

$$\forall M \in [M_0\rangle : \sum_{p \in P} W_p(M(p)) = \sum_{p \in P} W_p(M_0(p))$$

Jensen todistaa artikkelissa [32], että W:n ollessa paikkavuo, se myös määrittelee kaikissa tapauksissa paikkainvariantin. Sen sijaan mikäli W määrittelee paikkainvariantin, se on myös paikkavuo vain siinä tapauksessa, että väritetty Petri-verkko ei sisällä yhtään kuollutta sidonta-alkiota.

Määritelmässä 4.22 esitettiin kuinka paikkainvariantti voidaan määritellä paikkavuon avulla. Paikkavuon määritelmässä 4.21 käytettyjen painojen etsimiseen on kaksi tapaa. Ne voidaan etsiä joko automaattisesti tai suunnittelijan avustuksella.

Väritetty Petri-verkko voidaan esittää matriisimuodossa, kuten perinteisetkin Petri-verkot. Väritetyn Petri-verkon matriisi I ei kuitenkaan sisällä kokonaislukuja, vaan funktioita. Matriisi sisältää rivin jokaista paikkaa, ja sarakkeen jokaista siirtymää kohti. Samaan tapaan kuin perinteisten Petri-verkkojen kytkentämatriisissa, matriisin I alkio (p, t) kuvaa miten siirtymän t suorittaminen muuttaa paikan p

merkkausta. Tällöin matriisin alkio (p, t) on funktio, joka kuvaa jokaisen mahdollisen sidonnan $b \in B(t)$ merkkien painotetuksi joukoksi

$$E(t, p)\langle b \rangle - E(p, t)\langle b \rangle \text{ [32]}.$$

Olkoon rivivektori $W = \{W_p \mid p \in P\}$ painojen joukko. Määriteltäessä funktioiden kertolasku yhdistetyksi funktioksi, painojen joukko W on paikkavuo, mikäli [29]

$$W \cdot I = 0.$$

Väritetyn Petri-verkon paikkainvariantteja voidaan etsiä edellä esitetyn yhtälön avulla. Vastaavasti voidaan määrittää siirtymille painot ja luoda näiden avulla siirtymävuoita. Määriteltäessä rivivektori $W = \{W_t \mid t \in T\}$ siirtymien painojen joukoksi, on W siirtymävuo, mikäli [64]

$$W \cdot I^T = 0.$$

Paikka- ja siirtymäinvariantteja voidaan etsiä edellä esitettyjen yhtälöiden avulla. Tämä tapahtuu samaan tapaan kuin perinteisillä Petri-verkoilla, mutta kokonaislukumatriisien sijaan käsitellään funktioita sisältäviä matriiseja. Tämä voi olla erittäin työlästä, sillä kaikille funktioille ei välttämättä ole olemassa käänteisfunktioita, ja invarianttien määrä voi olla ääretön. Lisäksi löydetyt invariantit voivat olla järjestelmän analysoinnin kannalta täysin turhia.

Edellä esitellyistä syistä invarianttien automaattista etsimistä ei esitellä tarkemmin. Jensen kuvaa artikkelissaan [29] väritetyn Petri-verkon matriisimuodon sekä invarianttien etsimisen tässä esiteltyä tarkemmin. Silva ym. esittävät artikkelissaan [64] algoritmin väritetyn Petri-verkon perusvuoiden etsimiseen. Perusvuoiden avulla voidaan muodostaa lineaarikombinaatioina väritetyn Petri-verkon kaikki vuot.

Nykyään huomattavasti yleisempi ja käytännöllisempi tapa etsiä invariantteja on jättää niiden määritteleminen järjestelmän suunnittelijan vastuulle. Suunnittelijan tehtävänä on siis määritellä paikoille ja siirtymille painot. Monet näistä voivat olla nollafunktioita, jotka palauttavat tyhjän painotetun joukon kaikilla lähtöjoukon alkioilla. Painojen määrittelemisen jälkeen tehtäväksi jää todistaa, että kyseisten painojen joukko määrittelee vuon. Tämä voidaan tehdä työkalun avulla automaattisesti. Mikäli painojen joukko ei määrittele vuota, tulee suunnittelijan muokata mallia tai painoja siten, että järjestelmän spesifikaatioista löytyvät invariantit pätevät.

Väritetyn Petri-verkon paikka- ja siirtymäinvarianteilla voidaan analysoida luvussa 4.4.1 esitettyjä järjestelmän ominaisuuksia samaan tapaan kuin luvussa 3.4.3 esitettiin perinteisille Petri-verkoille. Invarianttien käyttämisessä järjestelmän analysoinnissa on useita etuja tapahtumagraafiin verrattuna. Tärkein etu on invarianttien mahdollistama hierarkkisten väritettyjen Petri-verkkojen tehokas analysointi. Tämä etu saadaan siitä, että yksittäisille sivuille määritellyistä invarianteista voidaan yhdistää koko hierarkkista verkkoa kattavia invariantteja [50].

Invarianttien etuna tapahtumagraafiin nähden on myös se, että järjestelmän parametreja ei tarvitse kiinnittää invarianttianalyysiä varten. Invarianteilla tehty analysointi pätee siis järjestelmän kaikilla parametreilla. Huonona puolena on se, että invarianttien käyttäminen vaatii suunnittelijalta paremmat ja matemaattisemmat taidot sekä enemmän työtä kuin automaattisesti tehtävä tapahtumagraafipohjainen analysointi.

5 Kongressinhallintajärjestelmän mallintaminen väritetyillä Petri-verkoilla

Luvussa esitellään Kongressinhallintajärjestelmän mallintaminen väritetyillä Petri-verkoilla. Ensin annetaan kuvaus mallinnukseen käytetystä ohjelmistosta ja laitteistosta, jonka jälkeen esitellään mallinnetut verkot selityksineen.

5.1 CPN Tools -sovellus ja käytetty laitteisto

Kongressinhallintajärjestelmän mallintamiseen, simulointiin ja analysointiin väritetyillä Petri-verkoilla käytettiin CPN Tools -sovellusta [11]. CPN Tools on Aarhusin yliopistossa toimivan CPN Groupin [10] kehittämä ja ylläpitämä sovellus väritettyjen Petri-verkkojen käsittelyyn. Sovellus on täysin ilmainen kaikenlaisessa käyttössä sisältäen myös kaupallisen käytön. CPN Tools ei kuitenkaan ole vapaa ohjelmisto, vaan sitä kehitetään keskitetysti CPN Groupin toimesta. Tällä hetkellä (1.4.2007) sovellukseen on hankittu noin 5000 lisenssiä 127:stä maasta.

CPN Tools perustuu Design/CPN -sovellukseen [15], jota kehitettiin 1980-luvun lopusta 1990-luvun loppuun asti Meta Software Corporationin ja CPN Groupin yhteistyönä. Design/CPN on kaupallinen sovellus, jonka kehittäminen on lopetettu, eikä sen käyttöön myydä enää uusia lisenssejä. 1990-luvun lopussa aloitettu CPN Tools -sovelluksen kehitys on korvannut Design/CPN-sovelluksen. Sovellukset ovat periaatteiltaan hyvin samanlaisia, ja monista 1990-luvulla julkaistuista artikkeleista, joissa Design/CPN-sovellusta on käytetty tutkimuksessa työkaluna, saa hyvän kuvan myös CPN Tools -sovelluksen toiminnasta.

CPN Tools käyttää CPN ML -ohjelmointikieltä väritetyn Petri-verkon lausekkeiden ja määrittelyiden kielenä. CPN ML -kielen avulla määritellään esimerkiksi käytetyt tyypit, muuttujat, paikkojen alkumerkkaus ja kaarilausekkeet. CPN ML -kieli on suositun funktionaalisen SML-ohjelmointikielen (Standard ML) [46] laajennus. SML-kielen avulla väritetyn Petri-verkon lausekkeissa voidaan hyödyntää SML-kieltä varten tehtyjä laajoja kirjastoja. CPN Tools sisältää vapaan SML/NJ (Standard ML of New Jersey) [65] kääntäjän, jonka avulla CPN ML -kieltä voidaan suorittaa. Myös CPN Tools -sovellus on toteutettu osittain SML-kielellä, lukuun otta-

matta graafisia osia, jotka on toteutettu oliopohjaisella BETA-kielellä [3].

CPN ML -laajennus helpottaa SML-kielen käyttöä väritetyissä Petri-verkoissa. Se lisää syntaktista sokeria SML-kieleen, helpottaen yleisimpien värijoukkojen, ja käytettyjen funktioiden määrittelyä. Lisäksi CPM ML mahdollistaa muuttujien tyyppien määrittelemisen, mikä ei ole mahdollista SML-kielessä. Viimeisenä muutoksena se mahdollistaa SML:n referenssimuuttujien käytön vain siirtymien suoritukseen kuuluvissa ohjelmalohkoissa.

CPN Tools on monipuolinen sovellus, joka sisältää ominaisuuksia erilaisia käyttötarkoituksia varten. CPN Toolsin syyskuussa 2006 julkaistu versio 2.2.0 sisältää muun muassa seuraavia ominaisuuksia edellä esitetyn CPN ML -kielen tuen lisäksi [11, 35]:

- Yhtenäinen graafinen käyttöliittymä.
- Yksinkertaisten, hierarkkisten ja ajastettujen väritettyjen Petri-verkkojen mallintaminen. Sama malli voidaan muuttaa yksinkertaisesta hierarkkiseksi, ja haluttaessa myöhemmin ajastetuksi.
- Väritettyjen Petri-verkkojen simulointi manuaalisesti askeleittain tai automaattisesti.
- Mallinnetun verkon ominaisuuksien analysointi täyden tapahtumagraafin avulla.
- Suorituskyvyn analysoiminen ajastettujen väritettyjen Petri-verkkojen avulla.
- Monitoreiden lisääminen simuloinnin avuksi sekä tila-avaruudessa tehtävät kyselyt.
- XML-tiedostomuoto.

CPN Tools -sovelluksesta puuttuu tuki invarianttipohjaiselle analyysille, eikä se sisällä mitään täyden tapahtumagraafin muodostamista tehostavaa menetelmää. Design/CPN sisälsi mahdollisuuden invarianttien käyttämiseen järjestelmän analysoimisessa sekä pyyhkäisyviivamenetelmän tila-avaruuteen pohjautuvassa analyysissä. Tämän takia on todennäköistä, että nämä ominaisuudet tulevat joskus tulevaisuudessa myös CPN Toolsiin.

CPN Tools valittiin Kongressinhallintajärjestelmän mallintamisen työkaluksi edellä esiteltyjen ominaisuuksien ansiosta. Kongressinhallintajärjestelmän analysoiminen toteutettiin simuloimalla ja tila-avaruuteen perustuvilla menetelmillä, minkä

takia invarianttipohjaisen analysoinnin puuttuminen sovelluksesta ei ollut esteenä sen käytölle. CPN Tools tuli tutuksi lukuisista kokeellisista artikkeleista, joissa sitä oli käytetty työkaluna. Tämä helpotti sovelluksen käytön aloittamista. Sovellusta varten on olemassa myös teknisen tuen sähköpostilista, jonka avulla mahdollisiin ongelmatilanteisiin sai nopeasti suoraan kehittäjiltä vastauksen.

CPN Toolsin viimeisin versio on 2.2.0, joka julkaistiin syyskuussa 2006. Tämä versio toimii sekä Windows XP/2000- että Linux-ympäristössä. CPN Tools on sisältänyt alusta asti tuen Windowsille, mutta Linux on ollut tuettuna vasta versiosta 2.0.0 lähtien. Tästä syystä Windows-version voidaan olettaa sisältävän myös vähemmän bugeja kuin Linux-versio. Sovellus ei vaadi laitteistolta suurta tehoa, mutta tila-avaruuden analysoinnin nopeus on suoraan riippuvainen käytettävissä olevasta prosessoritehosta ja muistista. Näiden lisäksi CPN Tools vaatii laitteistotasolla OpenGL:ää tukevan näytönohjaimen.

Kongressinhallintajärjestelmän mallintamiseen ja simulointipohjaiseen analysointiin käytettiin muutaman vuoden vanhaa laitteistoa, jolla CPN Tools toimi ongelmitta. Taulukossa 5.1 on esitetty käytetyn laitteiston tarkemmat tiedot. Laitteisto sisälsi sekä Windows XP- että Fedora Core 5 Linux-käyttöjärjestelmän. Linuxilla ilmenneiden näytönohjaimen suorituskykyongelmien vuoksi Kongressinhallintajärjestelmän mallintaminen ja simulointipohjainen analysointi toteutettiin Windows XP -käyttöjärjestelmässä.

Prosessori	AMD Athlon XP 2000+ 1,67 GHz
Muisti (RAM)	512 MT
Näytönohjain	NVIDIA GeForce 4 Ti 4200 128 MT
Käyttöjärjestelmä	MS Windows XP Home Edition SP 2
CPN Tools	Versio 2.2.0 Windowsille

Taulukko 5.1: Kongressinhallintajärjestelmän mallintamiseen ja simulointipohjaiseen analysoimiseen käytetty laitteisto ja ohjelmistot.

Edellä esitetty mallinnukseen ja simulointiin käytetty laitteisto ei ollut riittävän tehokas mallin tila-avaruuden analysoimiseen. Lähinnä muistin riittämättömyys hidasti tila-avaruuden luontia ja analysointia niin paljon, että tämä osuus päätettiin tehdä toisella laitteistolla. Mallin tila-avaruuden luontiin, ja sen pohjalta tehtyyn analysointiraporttiin käytetty laitteisto on esitetty taulukossa 5.2.

Prosessori	AMD Athlon XP 3000+ 2,04 GHz
Muisti (RAM)	1 GT
Käyttöjärjestelmä	Fedora Core Linux 6
CPN Tools	Versio 2.2.0 Linuxille

Taulukko 5.2: Kongressinhallintajärjestelmän mallin tila-avaruuden analysoimiseen käytetty laitteisto ja ohjelmistot.

5.2 Kongressinhallintajärjestelmän mallintaminen

Luvussa esitetään Kongressinhallintajärjestelmän abstraktien vastaanotto ja arviointi -prosessin mallintaminen väritetyillä Petri-verkoilla käyttäen CPN Tools -sovellusta. Ensin esitetään yleiset mallinnuksessa käytetyt periaatteet, jonka jälkeen käydään läpi mallinnetut sivut yksitellen.

5.2.1 Mallinnuksen tarkoitus sekä käytetyt periaatteet ja merkintätavat

Kongressinhallintajärjestelmän mallintamisessa keskityttiin abstraktien vastaanotto ja arviointi -prosessiin. Tämä on koko järjestelmän toiminnan ja luotettavuuden kannalta olennaisin osa, sillä prosessin aikana järjestelmällä voi olla toista tuhatta käyttäjää. Tästä syystä abstraktien vastaanotto ja arviointi -prosessi valittiin tarkemman mallintamisen kohteeksi. Mallin analysointia kuvataan tarkemmin luvussa 6.

Mallinnuksessa pyrittiin noudattamaan Kongressinhallintajärjestelmän suunnitteluperiaatteita, jotta järjestelmässä mahdollisesti ilmeneviä virheitä voitaisiin löytää. Malliin ei kuitenkaan otettu kaikkea Kongressinhallintajärjestelmässä olevaa tietoa, vaan pelkästään sellainen tieto mallinnettiin, jolla oli merkitystä abstraktien tilasiirtymien kannalta. Esimerkiksi abstraktit eivät sisällä otsikkoa tai itse abstraktin tekstiä.

Malli tehtiin ensisijaisesti Kongressinhallintajärjestelmän suunnitelmien pohjalta. Tarkemmin suunnittelemattomien kohtien yhteydessä jouduttiin tutkimaan järjestelmän ohjelmakoodista toteutusta. Näin pyrittiin parantamaan mallin ja sen toteutuksen yhdenmukaisuutta. Tällä tavalla ei kuitenkaan voida varmistua siitä, että malli vastaisi käytännön toteutusta. Tästä syystä kaikki havaitut ongelmat varmistettiin myös oikeassa tuotantoympäristössä olevassa Kongressinhallintajärjestelmässä.

Mallissa paikkojen merkkaus pyrittiin pitämään yksittäisten merkkien monijouk-

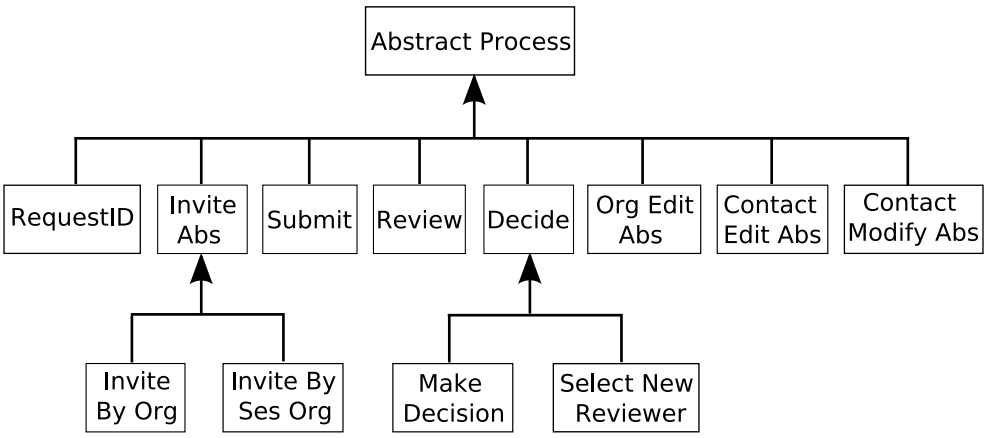
kona. Tietyissä tapauksissa täytyi paikan tyypiksi määritellä tietyn tyyppinen lista. Tällöin paikan merkkaus sisälsi monijoukon johon kuului yksi lista. Listojen käytöllä voitiin itse valita siirtymässä käytetty alkio sen sijaan, että olisi valittu jokin satunnainen paikan merkki. Lisäksi paikan ollessa tyhjä, voidaan kaarilausekkeessa olevaan muuttujaan sitoa tyhjä lista, jonka avulla siirtymä on mahdollinen. Ilman listojen käyttöä ei siirtymää voida suorittaa mikäli sen esijoukon paikassa ei ole yhtään merkkiä.

Esitetään seuraavaksi mallissa käytetyt merkintätavat. Esimerkkiä merkintätavoista voi katsoa kuvasta 5.3, joka sisältää monia käytettyjä merkintätapoja. Mallissa on paikan vieressä esitetty kyseisen paikan tyyppi kokonaan isoilla kirjaimilla. Käytetyt tyypit on määritelty liitteessä A.1. Paikkojen viereen on merkitty myös paikan alkumerkkaus siinä tapauksessa, että alkumerkkaus poikkeaa tyhjästä monijoukosta. Alkumerkkaus on yksinkertaisessa tapauksessa esitetty suoraan paikan vieressä tai sitten on käytetty erikseen määritettyä alustusvakiota. Alustusvakiot on esitetty kokonaan pienillä kirjaimilla, ja ne on määritelty liitteessä A.4.

Alkumerkkausten ja tyyppien lisäksi paikan viereen on voitu merkitä fuusiojoukko johon paikka kuuluu tai mahdollisen porttisolmun tyyppi. Nämä molemmat merkitään suorakaiteen muotoisella tekstilaatikolla. CPN Tools -sovelluksen bugista [12] johtuen paikka ei voi kuulua fuusiojoukkoon, ja olla samaan aikaan porttisolmu. Näin ollen paikan vieressä on enintään yksi suorakaiteen muotoinen tekstilaatikko. Mikäli paikka kuuluu fuusiojoukkoon, on kyseisen fuusiojoukon nimi tekstilaatikossa. Muussa tapauksessa tekstilaatikko sisältää porttisolmun tyypin *In*, *Out* tai *I/0*.

Kaarien yhteyteen on merkitty kaarilausekkeet. Kaarilausekkeet voivat sisältää muuttujia tai CPN ML -kielisiä lausekkeita. Kaarilausekkeissa olevat muuttujat on määritelty liitteessä A.2, ja käytetyt funktiot on määritelty liitteessä A.5. Joidenkin siirtymien yhteydessä on lisäksi esitetty vahtilauseke. Vahtilauseke on pyritty asettamaan riittävän havainnollisesti näkyviin siirtymän viereen. Kaari- ja vahtilausekkeissa on käytetty muutamaa vakiota, jotka on määritelty liitteessä A.3.

Abstraktien vastaanotto ja arviointi -prosessia mallinnettiin hierarkkisten väritettyjen Petri-verkkojen avulla. Kuvassa 5.1 esitetään mallinnettujen 13 sivun hierarkia. Sivujen hierarkia muodostettiin korvaavien siirtymien avulla. Korvaavat siirtymät on esitetty mallissa kaksinkertaisella reunaviivalla olevilla suorakulmioilla. Lisäksi korvaavien siirtymien yhteydessä esitetään mihin alisivuun korvaava siirtymä liittyy.



Kuva 5.1: Mallinnettujen verkkojen hierarkkinen rakenne.

Sivuilla käytettiin hierarkkisista menetelmistä lisäksi globaaleja ja sivufuusiojoukkoja. Taulukossa 5.3 on esitetty käytetyt fuusiojoukot. Fuusiojoukoista on esitetty nimen lisäksi tyyppi, alustuslauseke sekä sivut, joilla fuusiojoukkoon kuuluva paikka esiintyy. Kaikilla samaan fuusiojoukkoon kuuluvilla paikoilla on sama merkkaus.

5.2.2 Abstract Process -pääsivu

Mallin pääsivu on *Abstract Process*, joka kuvaa yleisellä tasolla koko abstraktien vastaanotto ja arviointi -prosessin. Pääsivu on esitetty kuvassa 5.2. Sivu sisältää kuusi paikkaa, jotka kuvaavat tiloja, joissa abstraktit voivat olla. Kaikkien pääsivulla olevien paikkojen tyyppi on ABSTRACT.

Pääsivulla on 14 korvaavaa siirtymää, jotka liittyvät kahdeksaan eri alisivuun. *Submit* alisivusta on kaksi instanssia ja *Org Edit Abs* -alisivusta kuusi erillistä instanssia. Näiden lisäksi sivut *Invite Abs* ja *Decide* sisältävät kaksi alisivua, jotka nähdään mallin hierarkkiakuvasta 5.1.

Pääsivulla ei ole yhtään tavallista siirtymää, mistä johtuen jokainen pääsivun paikka on socket-solmu kyseiseen paikkaan liitetyille alisivuille. Tästä syystä pääsivulla ei ole myöskään yhtään kaarilauseketta, jotka näkyvät muilla sivuilla kaarien yhteydessä.

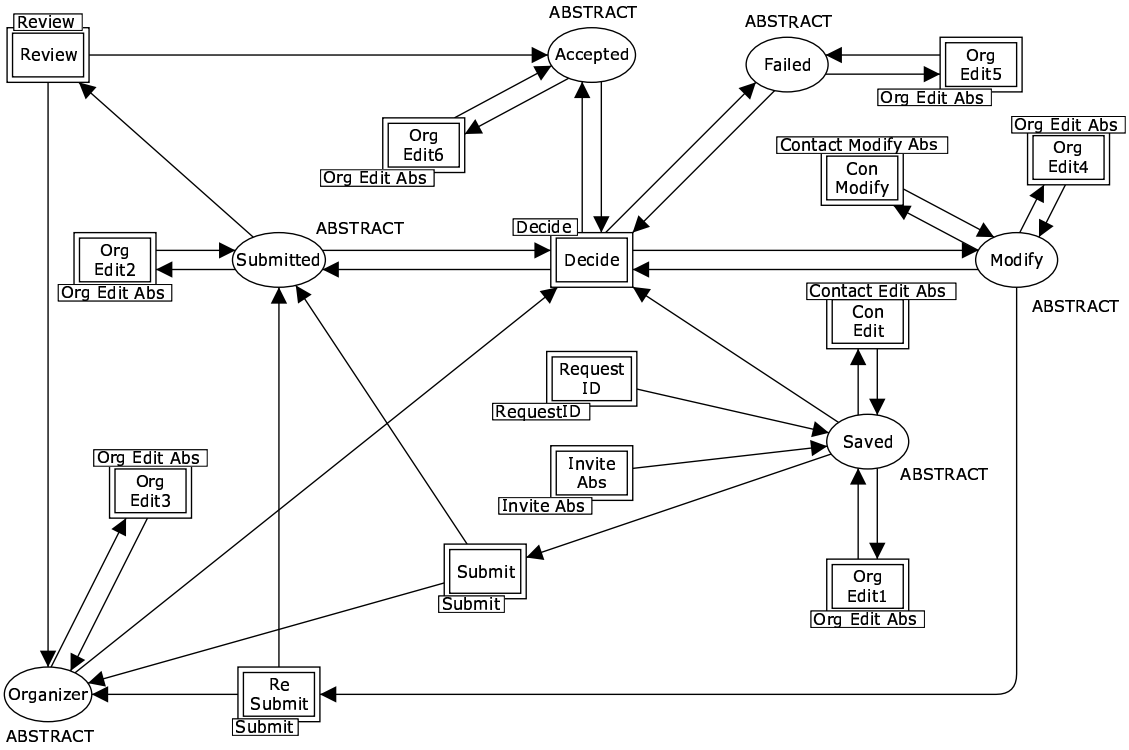
Pääsivun mallista 5.2 nähdään, että paikkaan *Saved* voi tulla abstrakteja kahdelta eri alisivulta. Alisivu *Invite Abs* kuvaa järjestäjän tai session järjestäjän kutsumia abstrakteja, ja *RequestID* yhteyshenkilöiden itsensä pyytämiä abstraktien tunnuk-

Nimi	Tyyppi	Alustus	Sivut
Fusion Abs_Rev	`ABSREVS`	`[]`	Review, Select New Reviewer ja Submit
Fusion Abs Count	`INT`	`0`	Invite By Org, Invite By Ses Org ja Request-ID
Fusion AIDs	`ID`	`init_abstractid`	Invite Abs ja Request-ID
Fusion Contacts	`USER`	`init_contacts`	Contact Edit Abs, Contact Modify Abs, Invite Abs ja Request-ID
Fusion Organizers	`ORGANIZER`	`init_organizers`	Invite By Org, Make Decision ja Org Edit Abs
Fusion Reviewers	`TOPS_REVS`	`init_tops_revs`	Select New Reviewer ja Submit
Fusion Sessiontypes	`SESSIONTYPES`	`init_stypes`	Contact Edit Abs, Org Edit Abs ja RequestID
Fusion Settings	`SETTINGS`	`init_settings`	Review ja Submit
Fusion SpecialCases	`SPECIALCASE`	`init_specialcases`	Contact Edit Abs ja Org Edit Abs
Fusion Topics	`TOPIC`	`init_topics`	Org Edit Abs ja RequestID

Taulukko 5.3: Kongressinhallintajärjestelmän mallissa esiintyvät fuusiojoukot.

sia. Paikassa *Saved* olevia abstrakteja voivat yhteyshenkilöt ja järjestäjät muokata. Mallista nähdään, että järjestäjät voivat muokata kaikissa paikoissa olevia abstrakteja. Yhteyshenkilöillä on oikeus muokata *Saved* paikan lisäksi vain paikassa *Modify* olevia abstrakteja. Session järjestäjän abstraktien muokkausoikeuksia ei ole mallinnettu, sillä abstrakteista mallinnettujen tietojen muokkausoikeudet ovat sessioiden järjestäjillä samat kuin yhteyshenkilöillä.

Paikassa *Saved* oleva abstrakti voi vaihtaa paikkaa joko jätettäessä abstrakti ar-



Kuva 5.2: Mallinnetun verkon pääsivu *Abstract Process*.

viointiin, jota on mallinnettu alisivulla *Submit*, tai järjestäjän päättämällä abstraktille uuden tilan, mikä on esitetty alisivulla *Decide*. Järjestäjällä on oikeus tehdä päätös abstraktin uudesta tilasta abstraktin nykyisestä tilasta riippumatta, minkä takia pääsivun jokaisesta paikasta on nuoli alisivulle *Decide*.

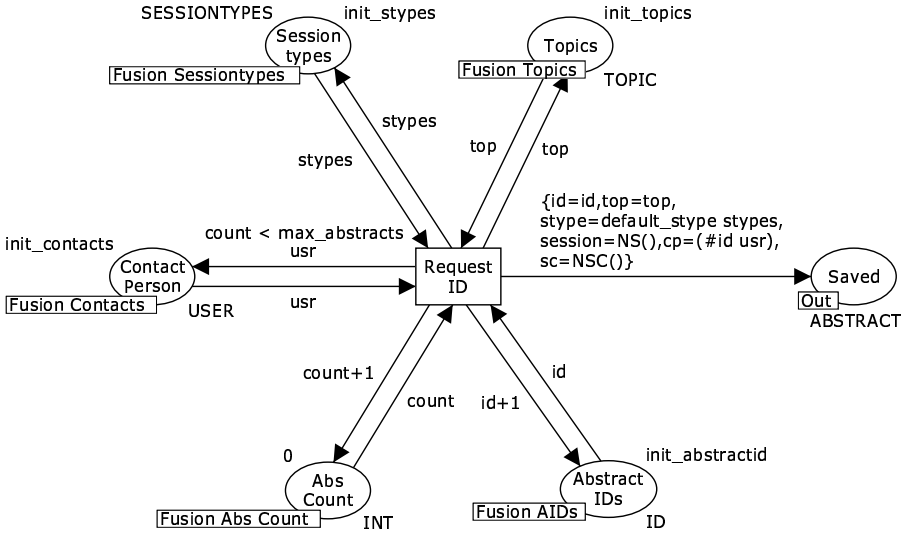
Paikka *Submitted* sisältää arvioijilla arvioinnissa olevat abstraktit. Aikaisemmin mainittujen ominaisuuksien lisäksi abstrakti voi poistua paikasta arvioinnin mukaan joko paikkaan *Organizer* tai *Accepted*. Arviointia on mallinnettu alisivulla *Review*. Paikassa *Organizer* olevat abstraktit odottavat järjestäjän toimenpiteitä. Tässä paikassa olevat abstraktit voivat siirtyä toisiin paikkoihin vain järjestäjän tekemän päätöksen avulla.

Paikka *Modify* sisältää yhteyshenkilöllä muokattavana olevat abstraktit. Yhteyshenkilö voi jättää muokattavana olevan abstraktin uudestaan arviointiin. Abstraktin tiedoista riippuen, se menee arviointiin joko arvioijalle tai järjestäjille. Tätä on mallinnettu korvaavalla siirtymällä *Re Submit*, joka on alisivun *Submit* toinen instanssi. Edellä mainittujen paikkojen lisäksi abstrakti voi olla paikassa *Failed* tai *Accepted*. Näissä paikoissa ovat hylätyt ja hyväksytyt abstraktit eikä niihin voi vaikuttaa ket-

kään muut kuin järjestäjät muokkaamalla tai tekemällä abstraktille uuden päätöksen.

5.2.3 RequestID-sivu

Kuvassa 5.3 on esitetty yhteyshenkilön hankkima abstraktin tunnistenumero. Sivu sisältää yhden siirtymän nimeltä *Request ID* ja kuusi paikkaa. Paikoista viisi kuuluu eri fuusiojoukkoihin. Paikka *Saved* on sivun tulosporttisolmu. Sivu on *Invite Abs* -sivun ja sen alisivujen lisäksi ainoa, jonka avulla malliin voidaan tuoda lisää abstrakteja.



Kuva 5.3: Mallinnetun verkon sivu *RequestID*.

Paikka *Contact Person* sisältää yhteyshenkilön, joka pyytää abstraktilleen id:tä järjestelmään. Tämän paikan merkkaus pysyy aina samana suoritettaessa siirtymä *RequestID*. Paikka *Session types* sisältää kongressiin asetetut sessiotyypit, ja paikka *Topics* kongressin aihealueet. Näidenkin molempien merkkaus pysyy muuttumattomana suoritettaessa sivun siirtymä.

Paikka *Abstract IDs* sisältää seuraavan pyydettävän abstraktin id-numeron. Paikassa olevaa kokonaislukua kasvatetaan yhdellä jokaisella siirtymän suorituksella. Edellä mainittujen lisäksi sivulla on paikka *Abs Count*, jonka avulla pidetään huoli siitä, että abstrakteja voidaan ottaa malliin vain haluttu lukumäärä. Abstraktien lukumäärän yläraja asetetaan vakiossa `max_abstracts`, jonka suuruutta paikan *Abs Count* merkkiin verrataan siirtymän vahtifunktiossa. Kongressinhallintajärjes-

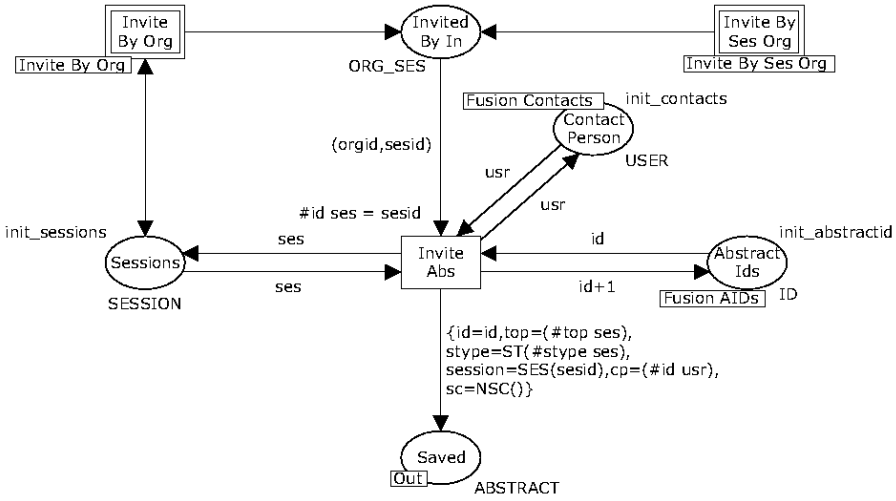
telmässä tällaista asetusta ei ole, mutta sen avulla voidaan mallin tila-avaruus pitää äärellisenä.

Suoritettaessa siirtymä *Request ID*, lisätään paikkaan *Saved* yksi uusi merkki tyypiltään `ABSTRACT`. Abstrakti saa seuraavana olevan id-numeron sekä satunnaisen aihealueen, jonka Kongressinhallintajärjestelmässä yhteyshenkilö valitsee. Abstraktin sessiotyypiksi asetetaan funktion `default_stype` palauttama sessiotyyppi tai `unit`-tyyppinen arvo. Funktion tehtävänä on etsiä järjestelmästä sessiotyyppiä, joka on asetettu abstraktien oletussessiotyypiksi. Mikäli tällaista ei ole, abstraktille ei aseteta sessiotyyppiä vaan `unit`-tyyppinen arvo.

Näiden lisäksi abstraktin sessioksi asetetaan `unit`-arvo, sillä abstraktia ei oletuksena liitetä mihinkään kongressin sessioon. Abstraktin muuttujan `cp` arvoksi asetetaan yhteyshenkilön id-numero. Lopuksi abstrakti liitetään tyhjään erikoistapaukseen. Erikoistapauksia kuvaa abstraktin muuttuja `sc`, johon asetetaan `unit`-tyyppinen arvo.

5.2.4 Invite Abs -sivu

Kuvassa 5.4 on esitettynä mallinnettu sivu *Invite Abs*. Sivulla mallinnetaan kutsuttujen abstraktien lisäämistä järjestelmään. Sivun avulla mallinnetaan sekä järjestäjien että session järjestäjien kutsumia abstrakteja. Näitä molempia varten on erikseen omat alisivunsa.



Kuva 5.4: Mallinnetun verkon sivu *Invite Abs*.

Sivulla on yksi siirtymä sekä viisi tähän yhdistettyä paikkaa. Alisivuista *Invite By Org* ja *Invite By Ses Org* saadaan sivulle lisää merkkejä socket-solmuun *Invited By In*, jossa olevat merkit sisältävät kutsuvan järjestäjän id-numeron sekä session, johon abstrakti kutsutaan.

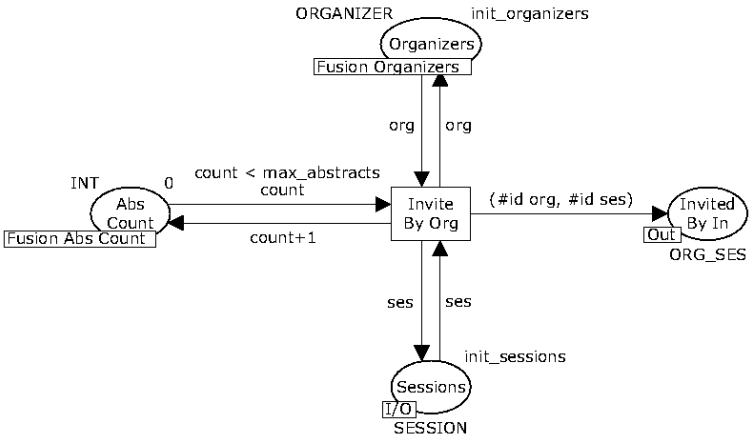
Paikka *Sessions* sisältää kongressiin luodut sessiot. Paikan ja siirtymän vahtifunktion avulla pidetään huoli siitä, ettei abstraktia voida kutsua sessioon, jota järjestelmässä ei ole. Siirtymän *Invite Abs* esijoukkoon kuuluvat lisäksi paikat *Contact Person* ja *Abstract IDs*, joista ensimmäinen sisältää yhteyshenkilöt ja jälkimmäinen abstraktien id-laskurin.

Suoritettaessa siirtymä *Invite Abs*, poistetaan paikasta *Invited By In* merkki. Samalla uuden abstraktin id-numero otetaan paikasta *Abstract IDs*. Uuden abstraktin yhteyshenkilön id-numero saadaan paikasta *Contact Person*. Siirtymän suoritus palauttaa otetut merkit paikkoihin *Sessions* ja *Contact Person* sekä asettaa abstraktin id:n laskuriin yhtä suuremman arvon. Samalla tulosporttisolmuun *Saved* lisätään uusi abstrakti, jonka id-numero on saatu laskurista. Abstraktin aihealue, sessiotyyppi sekä session id saadaan sessiolta, johon abstrakti kutsuttiin. Vastaavasti abstraktin yhteyshenkilön id saadaan valitulta yhteyshenkilöltä. Paikka *Saved* vastaa saman nimistä paikkaa pääsivulla.

5.2.5 Invite By Org -sivu

Invite By Org -sivun avulla mallinnetaan järjestäjän kutsumia abstrakteja. Sivua käytetään *Invite Abs* -sivun apuna, ja se on esitetty kuvassa 5.5. Sivun avulla päätetään kuka järjestäjistä haluaa kutsua abstraktin ja mihin sessioon. Sivu sisältää yhden siirtymän ja neljä paikkaa.

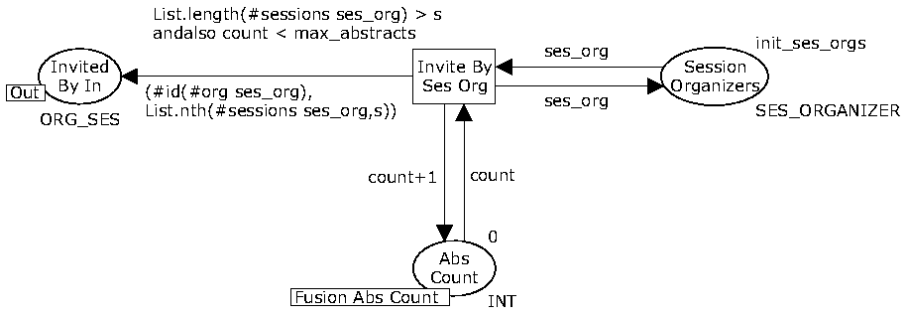
Suoritettaessa siirtymä *Invite By Org*, valitaan järjestäjä paikasta *Organizers* ja sessio paikasta *Sessions*. Molemmista paikoista otetut merkit palautetaan sellaisenaan takaisin paikkoihin. Siirtymän suorituksen aikana paikassa *Abs Count* olevaa abstraktien laskuria kasvatetaan yhdellä. Laskurin ja siirtymän vahtifunktion avulla pidetään huoli, ettei malliin voida lisätä rajattomasti abstrakteja. Siirtymän suorituksen tuloksena tulosporttisolmuun *Invited By In* lisätään pari, joka sisältää kutsuvan järjestäjän id-numeron sekä session id-numeron. Tämä paikka on yhdistetty sivun *Invite Abs* saman nimiseen paikkaan.



Kuva 5.5: Mallinnetun verkon sivu *Invite By Org*.

5.2.6 Invite By Ses Org -sivu

Sivun *Invite By Ses Org* avulla mallinnetaan session järjestäjän tekemää abstraktin kutsumista. Sivua käytetään *Invite Abs* -sivun apuna. Sivulla on yksi siirtymä ja kolme paikkaa, jotka on esitetty sivun mallissa kuvassa 5.6.



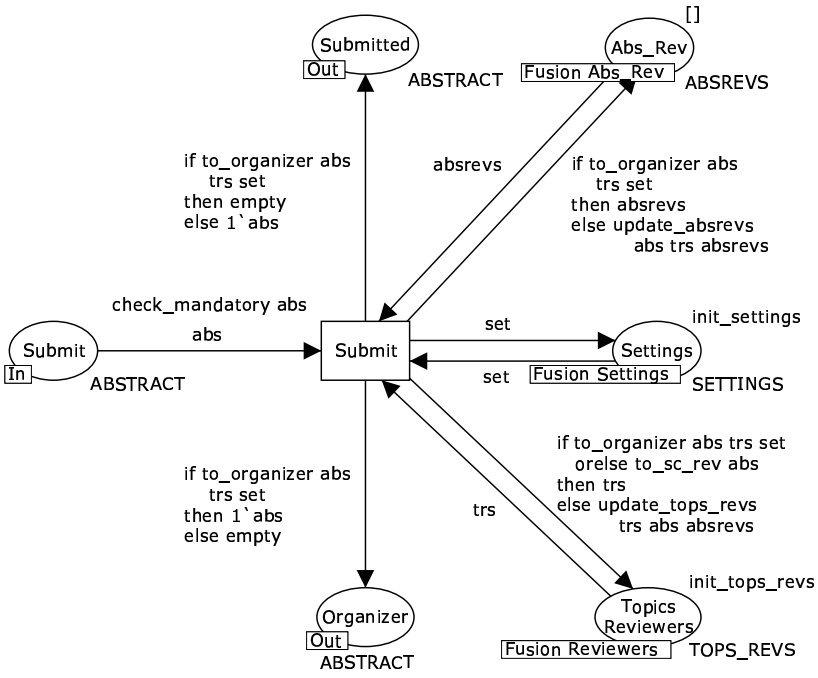
Kuva 5.6: Mallinnetun verkon sivu *Invite By Ses Org*.

Paikka *Session Organizers* sisältää kongressiin asetetut sessioiden järjestäjät. Session järjestäjien lisäksi paikassa on jokaisen järjestäjän hallitsemat sessiot. Siirtymän *Invite By Ses Org* suorituksen yhteydessä valitaan yksi session järjestäjä. Vahtifunktion avulla valitaan lisäksi yksi session järjestäjän hallitsemista sessioista, johon abstrakti halutaan kutsua. Vahtifunktion ja paikan *Abs Count* avulla pidetään huoli siitä, että malliin lisätään vain rajallinen määrä abstrakteja. Siirtymän suorituksen tuloksena tulosporttisolmuun *Invited By In* lisätään valitun järjestäjän id-numero sekä tämän hallitseman session id-numero. Paikkaan lisätty merkki näkyy myös sivun

Invite Abs saman nimisessä paikassa.

5.2.7 Submit-sivu

Kuvassa 5.7 esitetyn *Submit*-sivun avulla mallinnetaan abstraktin jättämistä arviointiin, sekä korjauksien jälkeen abstraktin jättämistä uudelleen arvioitavaksi. Arvioijan valintaprosessi on sen verran monimutkainen, että kaarilausekkeissa on käytetty paljon funktioita. Sivulla on yksi siirtymä ja kuusi paikkaa. Paikoista *Submit* on sivun syöteporttisolmu. Se sisältää samat merkit kuin pääsivun *Abstract Process* vastaava paikka. Paikat *Submitted* ja *Organizer* ovat sivun tulosporttisolmut, joihin lisätyt merkit näkyvät pääsivun vastaavissa paikoissa.



Kuva 5.7: Mallinnetun verkon sivu *Submit*.

Sivun tarkoituksena on siirtää arvioitavaksi jätetty abstrakti joko arvioijalle tai järjestäjille arvioitavaksi. Arvioijille menevät abstraktit lisätään paikkaan *Submitted*, ja järjestäjille menevät paikkaan *Organizer*.

Paikka *Settings* sisältää abstraktin arviointiin liittyvät kongressin asetukset. Näitä käytetään apuna päätettäessä kenelle abstrakti menee arvioitavaksi. Paikka *Topics Reviewers* sisältää aihealueiden arvioijat sekä tiedon siitä kenelle arvioijalle kyseisen aihealueen abstrakti viimeksi meni arvioitavaksi. Tämän avulla abstrakteja voidaan

jakaa tasaisesti kaikille aihealueen arvioijille. Paikassa *Abs_Rev* ylläpidetään listaa, jossa on kaikki arvioijilla arvioitavana olevat abstraktit sekä näiden arvioijat. Lisäksi listassa on tieto siitä, onko arvioija jo arvioinut abstraktin.

Siirtymän *Submit* suorittaminen on mahdollista vain sellaisille abstrakteille, joilla on syötettynä kaikki pakolliset tiedot. Tätä varten siirtymään on liitetty vahtifunktio. Mallin tapauksessa vahtifunktio tarkistaa vain, että abstraktille on asetettu sessiotyyppi, sillä se on ainoa pakollinen tieto mitä mallin abstrakteissa ei ole oletuksena.

Suoritettaessa siirtymä *Submit*, poistetaan abstrakti paikasta *Submit*, ja lisätään joko paikkaan *Submitted* tai *Organizer*. Abstrakti asetetaan arvioijalle arvioitavaksi mikäli arvioijien valitseminen on asetettu päälle, abstrakti ei kuulu sessioon, eikä sen sessiotyyppi ole kutsuttuja esityksiä varten. Lisäksi abstraktin aihealueelle täytyy olla asetettuna vähintään yksi arvioija tai abstraktin täytyy kuulua erikoistapaukseen, jolla on arvioija. Muissa tapauksissa abstrakti annetaan järjestäjille arvioitavaksi.

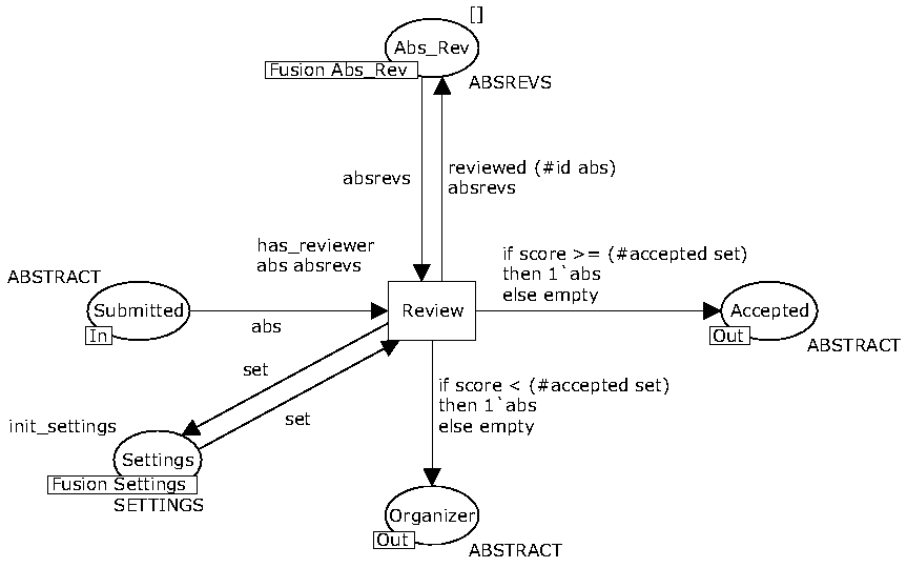
Paikan *Settings* merkkaus pysyy siirtymän *Submit* yhteydessä aina samana. Mikäli abstrakti asetetaan järjestäjille arvioitavaksi, pysyy myös paikkojen *Abs_Rev* ja *Topics Reviewers* merkkaus muuttumattomana. Lähetettäessä abstrakti arvioijalle arviointiin, päivitetään tieto tästä paikan *Abs_Rev* listaan. Samoin paikan *Topics Reviewers* listaan päivitetään kyseinen arvioija aihealueen viimeisimmäksi valituksi arvioijaksi, mikäli abstraktia ei asetettu erikoistapauksen arvioijalle arviointiin.

5.2.8 Review-sivu

Sivulla *Review* mallinnetaan arvioijan tekemää arviota abstraktista. *Review*-sivu esitetään kuvassa 5.8. Sivulla on yksi siirtymä ja viisi paikkaa, joista kolme on porttisolmuja. Syöteporttisolmu *Submitted* sisältää arviointia odottavat abstraktit. Arvioinnin perusteella abstraktit siirretään joko tulosporttisolmuun *Accepted* tai *Organizer*.

Paikka *Settings* sisältää kongressin hyväksymisrajan. Arvioinnissa vähintään asetuksista löytyvän arvosanan saaneet abstraktit hyväksytään suoraan kongressiin, ja siirretään paikkaan *Accepted*. Huonomman arvosanan saaneet abstraktit siirretään paikkaan *Organizer* odottamaan järjestäjien toimenpiteitä. Paikassa *Abs_Rev* on asetettuna jokaiselle arvioinnissa olevalle abstraktille arvioija. Tästä pidetään myös huolta siirtymän *Review* vahtifunktiossa, jonka avulla estetään ilman arvioijaa olevien abstraktien arviointi.

Suoritettaessa siirtymä *Review*, sidotaan arvosanaksi satunnainen arvo. Tämän



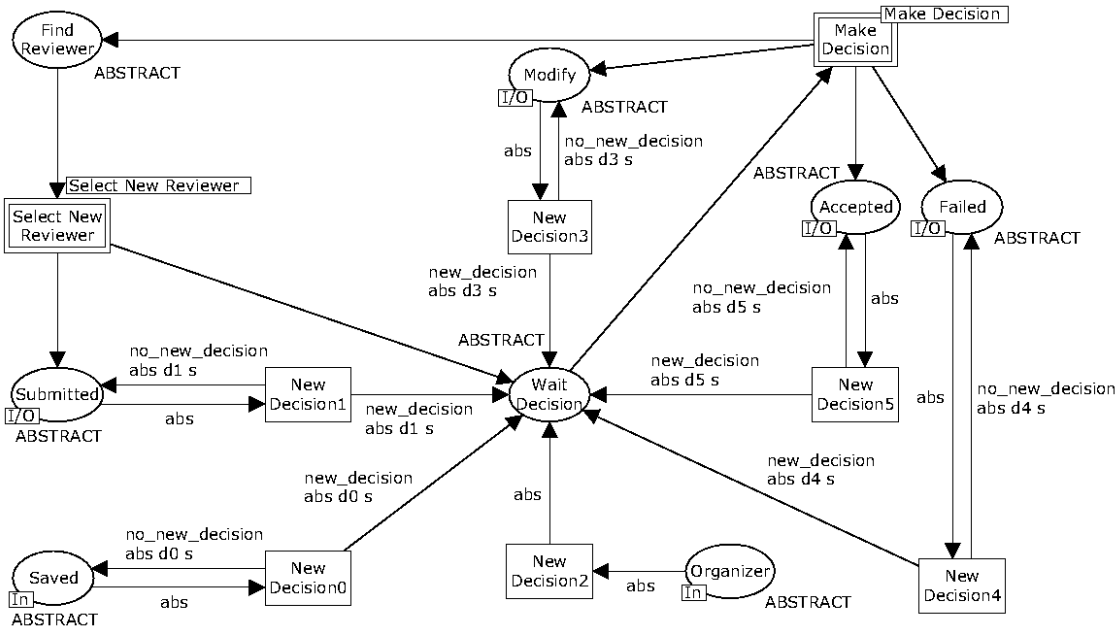
Kuva 5.8: Mallinnetun verkon sivu *Review*.

perusteella päätetään siirretäänkö paikasta *Submitted* poistettu merkki paikkaan *Accepted* vai *Organizer*. Lisäksi paikan *Abs_Rev* listaan päivitetään kyseinen abstrakti arvioiduksi.

5.2.9 Decide-sivu

Kuvassa 5.9 esitetyn *Decide*-sivun avulla mallinnetaan järjestäjän abstrakteille tekemiä päätöksiä. Kongressin järjestäjillä on rajattomat oikeudet abstraktien muuntamiseen tilasta toiseen. Tämä onnistuu neljän eri päätöksen avulla. Nämä ovat abstraktin määrääminen tietylle arvioijalle arviointiin, abstraktin asettaminen muokattavaksi, sekä abstraktin hyväksyminen tai hylkääminen. Normaalisti järjestäjä tekee näitä päätöksiä vain järjestäjälle arvioitavaksi lähetetyille abstrakteille tai sellaisille, jotka eivät ole saaneet arvioinnissa riittävän hyvää arvosanaa. Päätösten tekeminen on kuitenkin mahdollista missä tahansa tilassa oleville abstrakteille, mistä syystä nämä mahdollisuudet on lisätty kokonaisuudessaan tehtyyn malliin.

Sivulla on paikka jokaista abstraktin tilaa varten. Nämä kuusi paikkaa sisältävät saman merkkauksen kuin pääsivun *Abstract Process* vastaavat paikat. Näiden paikkojen lisäksi sivulla on kaksi apupaikkaa *Wait Decision* ja *Find Reviewer* abstrakteja varten. Sivun tarkoituksena on siirtää ne abstraktit paikkaan *Wait Decision*, joille järjestäjä haluaa tehdä uuden päätöksen. Tämä paikka toimii alisivun *Make Decision*



Kuva 5.9: Mallinnetun verkon sivu *Decide*.

socket-solmuna, jolla mallinnetaan järjestäjän tekemä päätös.

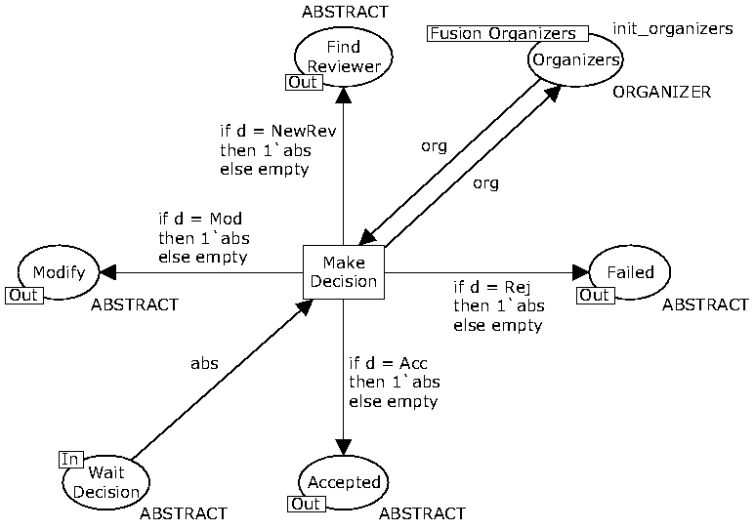
Jokaisesta abstraktin eri tilasta on siirtymä paikkaan *Wait Decision*. Lukuun ottamatta siirtymää paikasta *Organizer* paikkaan *Wait Decision*, siirtymät sisältävät satunnaisuutta. Tämä on mallinnettu sitä varten, etteivät mallia simuloitaessa niin monet abstraktit saisi jatkuvasti järjestäjiltä uutta päätöstä. Tarkoituksena on verrata satunnaista lukua ja malliin asetettua vakiota, ja tämän vertailun tuloksena joko palauttaa abstrakti lähtöpaikkaan tai lisätä se paikkaan *Wait Decision*.

Paikassa *Find Reviewer* olevat abstraktit odottavat uuden arvioijan asettamista. Paikka toimii socket-solmuna alisivulle *Select New Reviewer*, jolla uusi arvioija asetetaan. Mikäli kyseisellä alisivulla ei voida asettaa abstraktille uutta arvioijaa, se lisätään paikkaan *Wait Decision*. Muussa tapauksessa abstrakti asetetaan valitulle arvioijalle arviointiin paikkaan *Submitted*.

5.2.10 Make Decision -sivu

Sivulla *Make Decision* mallinnetaan järjestäjän tekemää päätöstä koskien abstraktia. Sivu on esitetty kuvassa 5.10. Päätöstä odottavat abstraktit ovat syöteporttisolmussa *Wait Decision*. Suoritettaessa siirtymä *Make Decision*, abstrakti siirretään yhteen neljästä tulosporttisolmusta. Nämä ovat *Modify*, *Find Reviewer*, *Failed* ja *Accepted*. Kaik-

ki porttisolmut liittyvät vastaaviin solmuihin ylisivulla *Decide*. Päätös on täysin satunnainen, ja se tehdään sitomalla siirtymän suorituksen aikana muuttujaan *d* yksi mahdollisista päätöksen arvoista. Siirtymän suoritusta varten tulee paikassa *Organizers* olla järjestäjä, joka tekee kyseisen päätöksen.



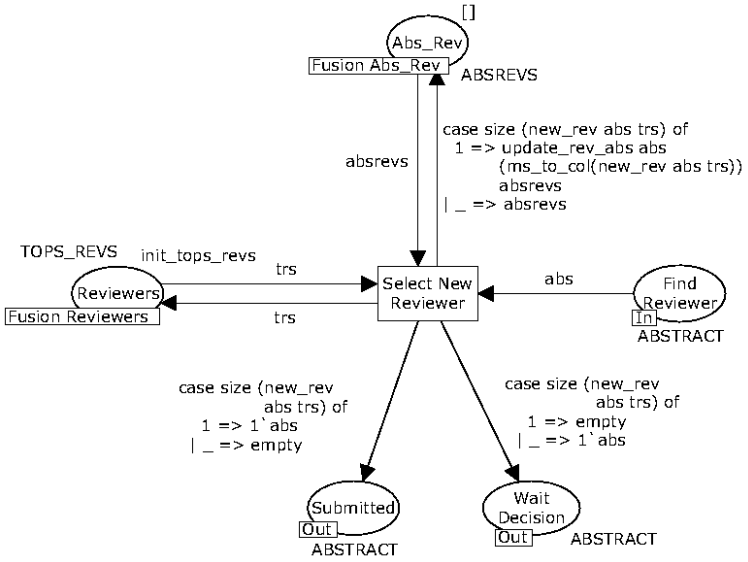
Kuva 5.10: Mallinnetun verkon sivu *Make Decision*.

5.2.11 Select New Reviewer -sivu

Kuvassa 5.11 on esitetty järjestäjän tekemä abstraktin uuden arvioijan valinta sivulla *Select New Reviewer*. Sivu sisältää yhden siirtymän ja viisi paikkaa. Syöteporttisolmussa *Find Reviewer* olevat abstraktit odottavat arvioijan valintaa. Siirtymän suorituksen tuloksena abstrakti siirtyy joko tulosporttisolmuun *Submitted* tai *Wait Decision*. Paikkaan *Submitted* siirretään ne abstraktit, joille löydetään arvioija. Mikäli arvioijaa ei löydy, abstrakti siirretään paikkaan *Wait Decision* odottamaan uutta päätöstä järjestäjiltä.

Sivun paikassa *Reviewers* ovat aihealuekohtaiset arvioijat. Tämä paikan listasta etsitään abstraktille uutta arvioijaa. Siirtymän *Select New Reviewer* suorituksen aikana listaa ei kuitenkaan päivitetä, jotta seuraavan aihealuekohtaisen arvioijan asettaminen sivulla *Submit* ei muuttuisi. Malli toimii tässä tapauksessa loogisesti täysin samaan tapaan kuin Kongressinhallintajärjestelmä.

Uuden arvioijan asettaminen, suorittamalla sivun siirtymä, päivittää paikassa *Abs_Rev* olevaa abstraktien arvioijien listaa. Kyseiseen listaan päivitetään abstrak-



Kuva 5.11: Mallinnetun verkon sivu *Select New Reviewer*.

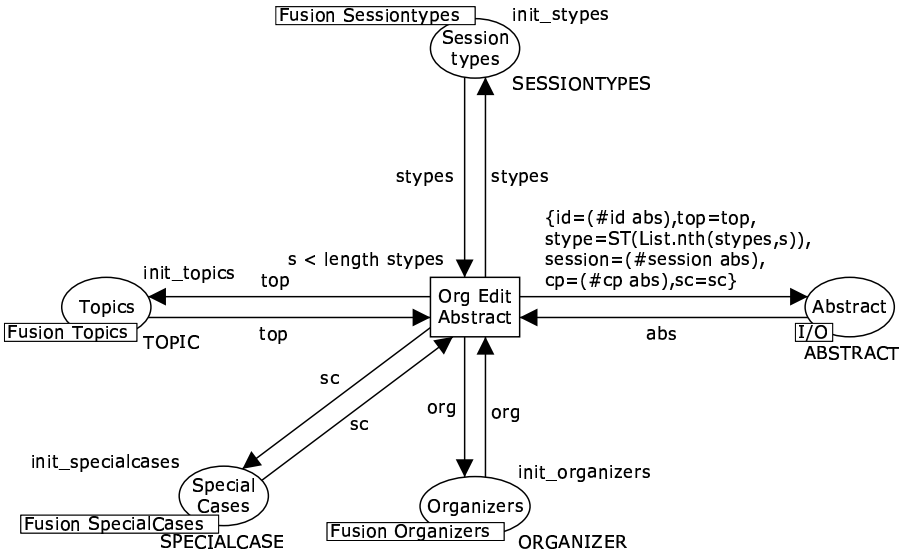
tille uusi arvioija, mikäli sellainen löytyy, ja asetetaan arvioinnin arvoksi `false`. Mikäli uutta arvioijaa ei löydy, paikan merkkaus pysyy muuttumattomana.

5.2.12 Org Edit Abs -sivu

Sivulla *Org Edit Abs* mallinnetaan järjestäjien tekemiä muutoksia abstrakteille. Sivu esitetään kuvassa 5.12. Sivusta on olemassa kuusi erillistä instanssia mallinnetussa verkossa. Erillisiä sivuja on käytetty sen takia, jotta mallin pääsivusta *Abstract Process* saataisiin selkeämmän näköinen.

Sivulla on yksi porttisolmu, joka on sekä syöte että tulosporttisolmu. Paikan nimi on *Abstract*, ja se sisältää muokattavat abstraktit. Sivun jokaisen eri instanssin paikka *Abstract* vastaa mallin pääsivun abstraktien eri tiloja kuvaavia paikkoja. Suoritettaessa siirtymä *Org Edit Abstract*, paikasta poistetaan abstrakti, jonka jälkeen siihen lisätään uusi abstrakti muokatuilla tiedoilla.

Sivun paikka *Organizers* sisältää kongressin järjestäjät. Kongressissa täytyy olla vähintään yksi järjestäjä, jotta abstraktin muokkaaminen olisi mahdollista. Vastaavasti paikassa *Topics* täytyy olla vähintään yksi aihealue sekä paikassa *Sessiontypes* lista, joka sisältää vähintään yhden sessiotyypin, jotta sivun suorittaminen olisi mahdollista. Paikan *SpecialCases* ei ole pakko sisältää erikoistapauksia, sillä riittää sisällyttää paikkaan `unit`-tyyppinen merkki.



Kuva 5.12: Mallinnetun verkon sivu *Org Edit Abs*.

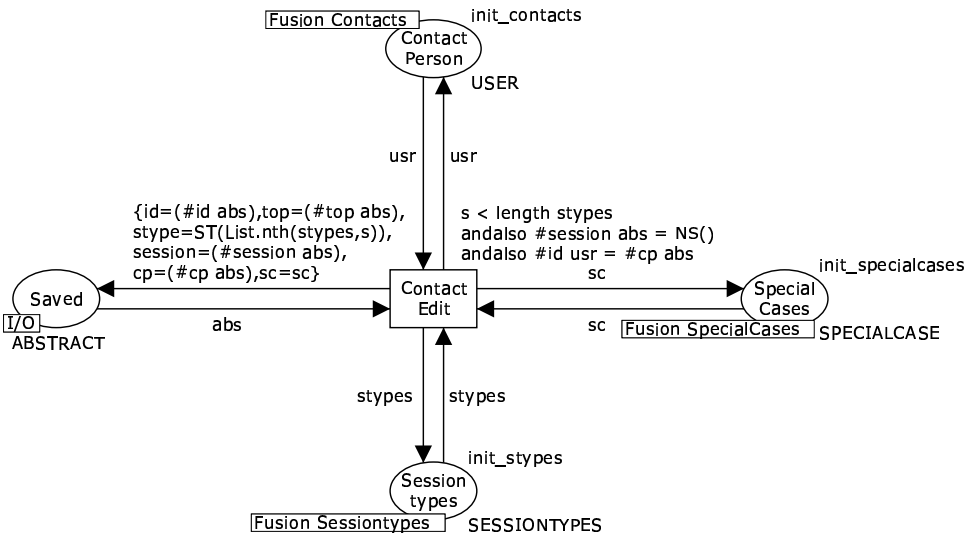
Sivun siirtymän suoritus ei muuta minkään muun paikan kuin *Abstract* merkkausta. Muista paikoista saadaan abstraktille uudet arvot. Siirtymän suorituksen yhteydessä abstraktille valitaan siis uusi satunnainen aihealue, sessiotyyppi ja erikoistapaus. Sen sijaan abstraktin id-numero, mahdollinen sessio, sekä yhteyshenkilön id pysyvät samoina.

5.2.13 Contact Edit Abs -sivu

Kuvassa 5.13 on esitettynä mallinnettu *Contact Edit Abs* -sivu. Sivun avulla mallinnetaan yhteyshenkilön abstraktin tietojen syöttämistä, kun abstrakti on tilassa *Saved*. Paikka *Saved* on myös sivun syöte ja tulosporttisolmu.

Paikka *Contact Person* sisältää kongressin yhteyshenkilöt. Sivulla olevan siirtymän *Contact Edit* vahtifunktiossa pidetään huoli siitä, että abstraktia voi muokata vain sen yhteyshenkilö. Paikka *SpecialCases* sisältää kongressin erikoistapaukset, ja paikka *Sessiontypes* kongressin sessiotyypit, kuten sivulla *Org Edit Abs*. Sivu ei kuitenkaan sisällä kongressin aihealueita, sillä yhteyshenkilöllä ei ole oikeutta muuttaa abstraktille valittua aihealuetta.

Siirtymän *Contact Edit* vahtifunktiossa tutkitaan, ettei abstraktia ole liitetty mihinkään sessioon. Mikäli abstrakti olisi kutsuttu sessioon, ei yhteyshenkilöllä olisi oikeuksia muuttaa sen sessiotyyppiä tai erikoistapausta. Siirtymän suorituksen ai-

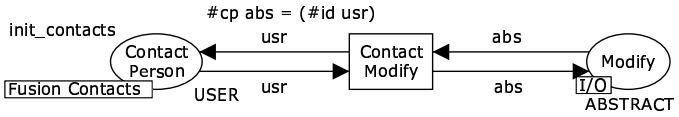


Kuva 5.13: Mallinnetun verkon sivu *Contact Edit Abs*.

kana abstraktille valitaan satunnainen sessiotyyppi ja erikoistapaus. Sen sijaan abstraktin id-numero, aihealue, session id ja yhteyshenkilö pysyvät samoina.

5.2.14 Contact Modify Abs -sivu

Sivulla *Contact Modify Abs* mallinnetaan yhteyshenkilön muokattavana olevaan abstraktiin tekemiä muutoksia. Sivu on esitetty kuvassa 5.14.



Kuva 5.14: Mallinnetun verkon sivu *Contact Modify Abs*.

Paikassa *Modify* ovat muokattavana olevat abstraktit. Tämä paikka on myös sivun syöte- ja tulosporttisolmu. Paikka *Contact Person* sisältää kongressin yhteyshenkilöt. Siirtymän *Contact Modify* vahtifunktiossa pidetään huoli siitä, että abstraktia voi muokata vain sen yhteyshenkilö. Siirtymän suorittaminen ei kuitenkaan muuta paikkojen merkkauksia ollenkaan. Tämä johtuu siitä, että abstraktin ollessa tilassa *Modify*, ei yhteyshenkilöllä ole oikeuksia muokata mallinnettuja abstraktin tietoja. Sivu mallinnettiin kuitenkin selkeämmän kuvan saamiseksi koko abstraktien vastaanotto ja arviointi -prosessista.

6 Mallinnettujen väritettyjen Petri-verkkojen analysointi

Kongressinhallintajärjestelmän abstraktien vastaanotto ja arviointi -prosessia on kehitetty parin vuoden aikana ilmenneiden tarpeiden mukaan. Prosessin suunnittelu on tapahtunut tänä aikana lähinnä järjestelmän suunnittelijoiden ajatuksissa. Tutkimuksessa rakennetut mallit ovat ensimmäisiä tarkempia kuvauksia prosessin toiminnasta. Tämän ansiosta prosessia on päästy tarkastelemaan uudesta näkökulmasta.

Analysoinnin aikana ei ollut mahdollista todistaa Kongressinhallintajärjestelmän oikeellista toimintaa, sillä mallin ja toteutuksen välisestä vastaavuudesta ei voitu olla varmoja. Sen sijaan tehdyt havainnot ja löydetyt virheet voitiin testata tuotantoympäristössä olevassa Kongressinhallintajärjestelmässä, ja näin varmistaa, että kyseessä eivät olleet pelkästään mallinnusvirheet.

Luvussa 5 esitettyjen mallien rakentamisen ja simuloinnin yhteydessä tuli esille uusia näkökulmia abstraktien vastaanotto ja arviointi -prosessiin, joita ei aikaisemmin oltu tutkittu. Tämä havainnollistaa sitä kuinka tärkeätä on suunnittelun yhteydessä mallintaa järjestelmän vähänkään monimutkaisemmat kokonaisuudet. Luvussa esitetään mallin analysointi sekä simuloimalla että tila-avaruuden avulla.

6.1 Mallin analysoiminen simuloimalla

CPN Tools -sovellus tarjoaa mahdollisuudet simuloida luotua mallia usealla eri tavalla. Erittäin hyödylliseksi todettiin simulointitapa, jossa mallia voitiin suorittaa askel kerrallaan, valitsemalla itse muuttujien sidonnat. Tämän avulla luodussa mallissa olleet virheet löytyivät helposti, ja samalla koko prosessia päästiin tarkastelemaan uudella tasolla. Tällä tavalla löytyneitä virheitä korjattiin edelleen, kunnes päädyttiin luvussa 5.2 esitettyihin malleihin. Mallin simuloinnin ja muokkausten yhteydessä kirjattiin ylös prosessissa mahdollisesti olevia ongelmakohtia.

Suurin osa mallinnuksen ja simuloinnin aikana ilmenneistä havainnoista olivat sen tyyppisiä, että niiden havaitseminen täysin staattisen analysointimenetelmän avulla olisi ollut hankalampaa. Näin Petri-verkkojen dynaamisuus osoittautui jär-

jestelmän analysoinnin kannalta erittäin hyödylliseksi.

Kongressinhallintajärjestelmän abstraktien vastaanotto ja arviointi -prosessin mallintamisen ja simuloinnin aikana tuli esille seuraavia aikaisemmin tekemättä jääneitä havaintoja:

1. Abstraktille voidaan määrätä arvioija, vaikka abstraktin sessiotyyppi olisi kutsuttuja esiintyjiä varten.

2. Abstrakti voidaan siirtää vain tiloihin *hyväksytty*, *hylätty*, *muokattavana* tai *arvioitavana* mistä tahansa muusta tilasta. Abstrakti tulisi voida siirtää myös tilaan *järjestäjällä*.

3. Asetetaanko abstrakti arviointiin uudelle vai edelliselle arvioijalle lähetettäessä abstrakti uudelleen arviointiin, mikäli järjestäjä on muuttanut abstraktin aihealuetta tai sessiotyyppiä?

4. Järjestäjä voi asettaa arvioijan abstraktille, joka on jo asetettu sessioon.

5. Miten erikoistapauksiin kuuluvien, kutsuttujen ja sessioihin asetettujen abstraktien arviointi menee abstraktin kuuluessa useampaan edellä mainittuun ryhmään?

6. Sessioon kuuluvaan abstraktiin ei voi session järjestäjä eikä yhteyshenkilö lisätä erikoistapauksia.

Listassa ensimmäisenä oleva ominaisuus ei ole suunniteltu, mutta se on järjestelmän monikäyttöisyyden kannalta kätevä ominaisuus. Vaikka kutsutut abstraktit tulevatkin suoraan järjestäjille arvioitavaksi, voidaan tietyissä tapauksissa haluta asettaa ne myös arvioijien arvioitavaksi.

Mallinnuksen yhteydessä tuli esille abstraktien tilasiirtymien kannalta puutteellinen toiminnallisuus. Järjestäjällä on oikeus hyväksyä, hylätä, lähettää muokattavaksi, tai asettaa uudelleen arviointiin missä tahansa tilassa oleva abstrakti. Näin ollen järjestäjä voi siirtää abstraktin mihin tahansa kuudesta tilasta lukuun ottamatta tiloja *tallennettu* ja *järjestäjällä*.

Tilaan *tallennettu* ei yhtään abstraktia tulekaan erikseen siirtää, mutta tilaan *järjestäjällä* tulisi missä tahansa tilassa oleva abstrakti voida siirtää. Näin järjestäjä voisi esimerkiksi perua tekemänsä virheellisen päätöksen siirtämällä abstrakti takaisin tilaan *järjestäjällä*. Tietyissä epäselvissä tapauksissa abstrakti voidaan myös haluta

siirtää tilaan *järjestäjällä* odottamaan järjestäjän päätöstä. Esimerkiksi tapauksessa, jossa sama henkilö olisi syöttänyt kaksi abstraktia kongressiin, vaikka vain yhden abstraktin syöttäminen olisi sallittua, voitaisiin nämä siirtää arvioijilta järjestäjille, ja ottaa kyseiseen henkilöön yhteyttä siitä, kumman abstraktin hän ennemmin haluaisi jättää kongressiin.

Mallin dynaamisuuden ansiosta simuloinnin aikana havaittiin toistaiseksi suunnittelematon kohta arviointiprosessissa. Tähän mennessä ei ole suunniteltu, kuinka tulisi toimia muokattavana olevien abstraktien kanssa, mikäli järjestäjä on vaihtanut niiden aihealuetta tai sessiotyyppiä. ECSS2007-kongressin yhteydessä toteutettiin ominaisuus, jonka avulla abstraktin nykyinen arvioija poistetaan, mikäli abstraktin sessiotyypiksi asetetaan kutsuttujen abstraktien sessiotyyppi. Näin ollen kutsutuiksi muutetut abstraktit tulevat muokkauksen jälkeen järjestäjille arvioitavaksi, kuten tulisikin tapahtua.

Aihealueen vaihdon yhteydessä muokattavana oleva abstrakti asetetaan oletuksena arviointiin entiselle arvioijalleen. Tietyissä tapauksissa tällainen toiminta on järkevää, sillä arvioija tuntee abstraktin entuudestaan. Mikäli abstraktin uusi aihealue ei kuitenkaan vastaa arvioijan osaamisaluetta, tulee abstraktille etsiä uuden aihealueen arvioija.

Mallinnuksen aikana huomattiin, että kongressin järjestäjä voi asettaa ulkoisesti hallittavaan sessioon kuuluvalle abstraktille uuden arvioijan. Kyseinen abstrakti ei kuitenkaan näy valitun arvioijan käyttöliittymässä, eikä näin ollen ole arvioijan arvioitavissa. Tässä toiminnallisuudessa on siis selvä bugi. Joko arvioijalle asetettu sessioon kuuluva abstrakti tulisi näkyä arvioijan käyttöliittymässä, tai sitten tällaiselle abstraktille ei tulisi edes voida määrittää uutta arvioijaa.

Mallinnuksen ja simuloinnin aikana huomattiin myös erikoistapauksiin liittyviä asioita, joita ei aikaisemmin oltu mietitty. Ensimmäisenä ilmennyt ongelma liittyi siihen kenelle abstrakti tulisi lähettää arviointiin, mikäli se kuuluu kutsuttujen abstraktien sessiotyyppiin, on asetettuna ulkopuolisesti hallittuun sessioon, ja sisältyy mahdollisesti vielä erikoistapaukseen.

Tämän hetken toteutuksessa abstrakti asetetaan ensisijaisesti arvioitavaksi session järjestäjille. Tämän jälkeen abstrakti asetetaan arvioitavaksi järjestäjille, mikäli se kuuluu kutsuttujen abstraktien sessiotyyppiin. Viimeisenä abstrakti annetaan mahdollisen erikoistapauksen arvioijalle. Tämä järjestys tuntuisi myös kaikista parhaalta, sillä ulkoisesti hallittavat sessiot olisivat joka tapauksessa ensisijaisessa asemassa. Lisäksi kutsutut abstraktit tulevat ennemmin järjestäjille arvioitavaksi kuin

erikoistapauksen arvioijalle. Tietyissä tapauksissa voisi olla mahdollista, että erikoistapauksen arvioija olisi järjestyksessä korkeammalla, mutta tarvetta tällaiselle ei ole vielä ilmennyt.

Viimeisenä havaintona mallinnuksen aikana huomattiin, että sessioon asetetulle abstraktille ei voi session järjestäjä eikä yhteyshenkilö asettaa erikoistapauksia. Tämä on selvä virhe järjestelmän toiminnassa, sillä erikoistapaukset voivat koskea mitä tahansa kongressiin liittyvää, ja myös ulkoisesti hallittujen sessioiden abstrakteilla tulisi olla oikeus osallistua niihin. Lisäksi edellä esitettiin, ettei erikoistapauksista koidu ongelmia ulkoisesti hallittuun sessioon kuuluvan abstraktin arviointiin.

6.2 Mallin analysoiminen tila-avaruuden avulla

CPN Tools -sovelluksessa voidaan käsitellä väritetyn Petri-verkon tila-avaruutta tätä varten suunnitellun työkalun [34] avulla. Tila-avaruuden työkalulla on mahdollista luoda mallin tila-avaruus sekä SCC-graafi. Luotua tila-avaruutta voidaan tutkia graafisesti, ja tietty tila on mahdollista siirtää malliin simulointia varten. Työkalun avulla voidaan myös luoda raportti mallin ominaisuuksista sisältäen mallin rajoittuvuus-, kotitila-, elävyys- ja reiluusominaisuudet.

Kongressinhallintajärjestelmän abstraktien vastaanotto ja arviointi -prosessia analysoitiin tila-avaruuden avulla. Mallin tila-avaruutta pyrittiin ensin analysoimaan mallinnuskoneella, joka on esitetty taulukossa 5.1. Täyden tila-avaruuden analysointi vei kuitenkin mallinnuskoneesta kaiken muistin, mikä hidasti toimenpidettä niin paljon, että täyden tila-avaruuden analysointi ei ollut enää järkevää mallinnuskoneella.

Tila-avaruuden räjähdyksen takia mallin alustuksia asetettiin sellaisiin arvoihin, joilla tila-avaruus olisi mahdollisimman pieni. Näin ollen päädyttiin analysoimaan vain yhden abstraktin tilasiirtymiä. Tämän takia menetettiin mahdollisuus analysoida mallissa olevia abstraktien keskinäisiä riippuvuuksia, joita ei tulisi olla. Lopulta mallin täysi tila-avaruus onnistuttiin luomaan, ja analysoimaan taulukossa 5.2 esitetyllä laitteistolla.

CPN Tools -sovelluksen luoma tila-avaruuden raportti on esitetty liitteessä B. Liitteessä B.1 on esitetty luotujen tila-avaruuden ja SCC-graafin tilastotiedot. Tila-avaruuden solmujen määrä pysyy kohtuullisena, sillä suurin osa tilojen eroista on abstrakteista riippuvaisia. Mallin yksi abstrakti ei kuitenkaan voi sisältää kuin 144 erilaista arvojen kombinaatiota, joten myös solmujen määrä pysyy äärellisenä. Sen

sijaan tila-avaruuden kaarien lukumäärä kasvaa erittäin suureksi mallin jokaisessa tilassa olleen järjestäjien muokkausoikeuden takia.

Mallin alustuksissa päätettiin jättää järjestäjältä pois mahdollisuus tehdä uusi päätös tiloissa *hyväksytty*, *hylätty* ja *tallennettu* oleville abstrakteille, jotta analysointia voitiin tehostaa. Tällä ei ollut analysoinnin kannalta suurta merkitystä, sillä yleensä tarkoituksena on, että järjestäjä ei tee tilamuutoksia kyseisissä tiloissa oleville abstrakteille.

Liitteessä B.2 esitetään mallin rajoittuvuusominaisuudet. Rajoittuvuusominaisuudet sisältävät sekä merkkien lukumäärien että monijoukkojen ylä- ja alarajat. Rajoittuvuusominaisuuksista nähdään, että abstraktin on mahdollista päätyä mihin tahansa mallinnetuista kuudesta eri tilasta. Lisäksi abstrakti voi sisältää minkä tahansa 144:stä eri arvokombinaatiosta näissä kaikissa tiloissa. Tämä on Kongressinhallintajärjestelmän monikäyttöisyyden kannalta tärkeä ominaisuus.

Kokonaislukurajoista nähdään, että malli ei vahingossa menetä kongressiin alustettua tietoa, kuten aihealueita. Monijoukon rajoista nähdään tarkempaa tietoa mallin suorituksen aikaisista merkkauksista. Paikan *Invited_By_In* monijoukon ylärajasta on nähtävissä, että järjestäjällä on oikeus kutsua abstrakti mihin tahansa sessioon, mutta session järjestäjä voi kutsua abstraktin vain omiin sessioihinsa.

Paikan *Abs_Rev* monijoukon ylärajasta on pääteltävissä, että abstrakti voidaan asettaa tiedoista riippuen mille tahansa arvioijalle. Lisäksi jokainen näistä arvioijista voi myös abstraktinsa arvioida. Paikan *Reviewers* monijoukon ylärajan avulla voidaan todistaa, että arvioijien aihealuekohtainen valinta toimii mallissa. Tämä nähdään siitä, että jokaisella aihealueella voi olla maksimissaan yksi viimeisin arvioija kerrallaan.

Liitteessä B.3 on esitetty mallista analysoidut kotitilaominaisuudet. Malli ei sisällä yhtään kotitilaa tila-avaruuden aikaisilla alustuksilla. Tämä on myös Kongressinhallintajärjestelmältä toivottava ominaisuus, vaikka abstrakteille tehdyt päätökset tulee voida perua. Kuitenkaan koko järjestelmän ei tule sisältää tilaa, johon voitaisiin aina kaikista tiloista päästä.

Mallista analysoidut elävyysominaisuudet on esitetty liitteessä B.4. Malli ei sisällä yhtään kuollutta merkkausta, joten kaikissa tiloissa on vähintään yksi mahdollinen siirtymä. Tämä todistaa, että suunniteltu malli ei voi joutua deadlock-tilaan ainakaan käytetyillä alustuksilla. Malli ei myöskään sisällä yhtään siirtymää, jota ei missään vaiheessa voitaisi myös suorittaa. Nämä molemmat ovat abstraktien vastaanotto ja arviointi -prosessin kannalta olennaisia ominaisuuksia. Suurin syy tähän

on se, että järjestäjillä on oikeus muokata ja tehdä uusia päätöksiä myös hyväksytyille ja hylätyille abstrakteille. Tämä ominaisuus antaa järjestäjille täyden kontrollin kongressin abstrakteihin.

Malli ei sisällä, tila-avaruuden analysoinnin aikaisilla alustuksilla, yhtään elävää siirtymää. Kongressinhallintajärjestelmässä järjestäjä voi kuitenkin tehdä uuden päätöksen missä tahansa tilassa olevalle abstraktille. Näin ollen monet *Abstract Process* -sivun, ja sen alisivujen, sekä *Org Edit Abs* ja *Contact Modify Abs* -sivujen siirtymät ovat itse asiassa eläviä. Tämä tieto menetettiin analysointiraportista, kun järjestäjältä poistettiin mahdollisuus tehdä uusi päätös tiloissa *hyväksytty* ja *hylätty* oleville abstrakteille.

Liitteessä B.5 esitetään mallista analysoidut siirtymien reiluusominaisuudet. Yksikään siirtymistä ei ole tasapuolinen, sillä mallissa on jokaista siirtymää kohti olemassa ääretön tapahtumasarja, jonka aikana kyseistä siirtymää ei suoriteta. Tämä on hyvä ominaisuus myös Kongressinhallintajärjestelmän kannalta. Näin Kongressinhallintajärjestelmä ei sisällä yhtään tapahtumaa, joka tapahtuisi kaikissa tapauksissa äärettömän useasti.

Siirtymät *Invite_Abs*, *Invite_By_Org*, *Invite_By_Ses_Org* ja *Request_ID* ovat reiluja, sillä nämä siirtymät ovat mahdollisia vain yhden kerran, jolloin ne myös suoritetaan. Lisäksi siirtymät *Make_Decision* ja *Select_New_Reviewer* ovat reiluja. Tämä johtuu siitä, että näiden siirtymien ollessa äärettömässä tapahtumasarjassa äärettömän useasti mahdollisia, niin ne myös suoritetaan äärettömän useasti. Näin mallin tuleekin toimia, sillä abstrakti siirretään pois normaaleista tiloistaan odottamaan näiden siirtymien suoritusta.

Siirtymä *Org_Edit_Abstract 3* on ainoa oikeudenmukainen siirtymä. Näin ollen tietystä hetkestä lähtien äärettömästi mahdollisena, siirtymä myös suoritetaan äärettömän monta kertaa. Tämä johtuu siitä, että *Decide*-sivulla ei mallinneta satunnaisuutta paikassa *Organizer* oleville abstrakteille tehtäville päätöksille. Näin ollen siirtymän *Org_Edit_Abstract 3* ollessa äärettömästi mahdollinen, ei siirtymän esipaikassa oleville abstrakteille voida suorittaa mitään muuta siirtymää. Muut mallin siirtymät eivät sisällä reiluutta.

7 Yhteenveto

Tutkimuksessa analysoitiin Jyväskylän yliopiston tietotekniikan laitoksella kehitetyn Kongressinhallintajärjestelmän abstraktien vastaanotto ja arviointi -prosessia. Tämä on Kongressinhallintajärjestelmän tärkein osa, sillä kongressista riippuen prosessilla on jopa toista tuhatta käyttäjää, ja suurin osa kongressin tiedoista kerätään sen aikana. Abstraktien vastaanotto ja arviointi -prosessia ei ole kuitenkaan missään vaiheessa mallinnettu ja analysoitu tarkemmin. Analysoinnin tarkoituksena oli etsiä prosessin suunnitelmassa mahdollisesti ilmeneviä virheitä, ja näin pyrkiä varmentamaan suunnitelman oikeellista toimintaa. Kongressinhallintajärjestelmän abstraktien vastaanotto ja arviointi -prosessia mallinnettiin ja analysoitiin väritettyjen Petri-verkkojen avulla.

Petri-verkot on yleisnimitys joukolle verkkoja, joilla on yhtäläinen graafinen esitysmuoto sekä vahva teoreettinen perusta. Graafisen esitysmuodon ansiosta Petri-verkkojen avulla voidaan mallintaa järjestelmiä havainnollisesti. Samalla mallinnettu verkko tarjoaa formaalin semantiikan ansiosta hyvät analysointimahdollisuudet. Petri-verkkojen vahvuutena ovat varsinkin niiden tarjoamat dynaamisten ominaisuuksien analysointimahdollisuudet. Petri-verkkojen dynaamisuuden ansiosta ne valittiin tutkimuksen mallinnus- ja analysointimenetelmäksi, sillä Kongressinhallintajärjestelmän abstraktien vastaanotto ja arviointi -prosessi on luonteeltaan dynaaminen.

Väritetyt Petri-verkot ovat korkean tason Petri-verkkoja, joille pätevät samanlaiset perusteet kuin muillekin Petri-verkoille. Ne eroavat perinteisistä Petri-verkoista tarjoamansa tyyppijärjestelmän ansiosta. Tämän takia suurin osa nykyisin mallinnettavista Petri-verkoista toteutetaan korkean tason Petri-verkoilla. Myös tässä tutkimuksessa käytettiin väritettyjä Petri-verkkoja niiden tarjoaman tyyppijärjestelmän ja kohtuullisen hyvän työkalutuen ansiosta.

Väritetyt Petri-verkot ovat kehittyneet ensimmäisten Petri-verkkojen matemaattisista yksityiskohdista kohti helppokäyttöistä suunnittelumenetelmää, joka edelleen sisältää Petri-verkkojen teoreettiset vahvuudet. Tästä syystä väritettyjen Petri-verkkojen käyttö ohjelmistokehityksessä on jatkuvasti kasvanut, ja niitä on käytetty menestyksellisesti yhä erilaisempien järjestelmien analysoimiseen.

Väritettyä Petri-verkkoa voidaan analysoida simuloimalla. Tämän lisäksi väritetyille Petri-verkoille on kehitetty lukuisia analysointimenetelmiä. Yleisimmät analysointimenetelmät perustuvat joko verkon tila-avaruuteen tai invariantteihin. Invarianttipohjainen analysointi vaatii suunnittelijalta yleensä enemmän työtä ja asiantuntemusta kuin tila-avaruuteen pohjautuvat menetelmät, jotka ovat täysin automaattisia. Tila-avaruuden analysoinnissa ongelmaksi muodostuu tilojen määrän räjähdys, joka estää laajojen järjestelmien tila-avaruuden täydellisen analysoinnin.

Kongressinhallintajärjestelmän abstraktien vastaanotto ja arviointi -prosessi mallinnettiin käyttäen CPN Tools -sovellusta. Tutkimuksessa tehty malli tarjoaa uuden näkökulman kehitettyyn järjestelmään. Tästä on hyötyä myös tulevaisuudessa kehitettäessä järjestelmää. Mallia analysoitiin sekä simuloimalla että mallin tila-avaruuden avulla. Simuloinnin aikana luodusta mallista tehtiin havaintoja, ja näitä kirjattiin ylös. Tila-avaruuteen pohjautuva analysointi toteutettiin CPN Tools -sovelluksella täysin automaattisesti, ja itse analysointi tehtiin sovelluksen tuottaman raportin pohjalta.

Järjestelmän analysoinnin aikana havaittiin muutamia ongelmakohtia abstraktien vastaanotto ja arviointi -prosessissa. Nämä virheet ja puutteet, sekä aikaisemmin miettimättä jääneet kohdat, havaittiin mallin simuloinnin yhteydessä. Virheellisenä ominaisuutena huomattiin, että järjestäjän on mahdollista asettaa ulkoisesti hallittavassa sessiossa olevalle abstraktille arvioija, mikä ei ole näiden sessioiden kannalta hyvä asia. Vaikka tämä ominaisuus haluttaisiinkin säästää, ilmenee järjestelmässä virhe, mikä estää kyseistä arvioijaa arvioimasta ulkoiseen sessioon kuuluvan abstraktin.

Hyödylliseksi ominaisuudeksi havaittiin mahdollisuus siirtää missä tahansa tilassa oleva abstrakti takaisin järjestäjien arvioitavaksi. Tällaista ominaisuutta ei järjestelmään ole kuitenkaan kehitetty. Simuloinnin aikana havaittiin myös, että sessioon kuuluvalle abstraktille ei voida lisätä erikoistapauksia muuta kuin kongressin järjestäjien toimesta. Muut simuloinnissa tehdyt havainnot olivat sellaisia, joita ei aikaisemmin ollut mietitty, mutta ne eivät sisältäneet puutteellisia toiminnallisuuksia.

Tila-avaruuden analysoinnin yhteydessä ongelmaksi oli muodostua tila-avaruuden kasvu liian laajaksi. Lopulta täysi tapahtumagraafi pystyttiin luomaan rajoittamalla mallin alustuksia. Mallin tila-avaruuden avulla pystyttiin todistamaan mallin toimintojen oikeellisuus. Nämä todistukset pätevät mallille käytetyillä alustuksilla, mutta niiden vastaavuudesta oikean toteutuksen kanssa ei voida varmistua.

Mallista havaittu olennainen tieto on se, että abstrakti voi kuulua mihin tahansa tilaan riippumatta sen sisältämistä tiedoista. Samalla voitiin todistaa aihealuekohtaisten arvioijien valinnan oikeellinen toiminta. Mallinnettu verkko ei sisältänyt yhtään kotitilaa eikä kuollutta merkkausta. Abstraktien vastaanotto ja arviointi -prosessin ei tulisikaan sisältää yhtään kotitilaa. Vastaavasti prosessin toiminnan kannalta on tärkeää, ettei se sisällä yhtään kuollutta merkkausta. Näin järjestelmä ei voi joutua deadlock-tilaan. Malli ei sisältänyt yhtään elävää siirtymää, vaikka järjestäjän tekemä uusi päätös tulisi olla aina mahdollinen. Tästä jouduttiin kuitenkin luopumaan ennen tila-avaruuden analysointia, jotta analysoinnin tila- ja aikavaativuutta pystyttiin pienentämään.

Tutkimuksessa tehdyt analysointitulokset antoivat hyviä havaintoja abstraktien vastaanotto ja arviointi -prosessin toiminnasta. Varsinkin simuloinnin aikana tehdyt havainnot olivat hyödyllisiä prosessin jatkokehityksen kannalta. Simuloinnin yhteydessä havaitut ongelmat pystyttiin myös todentamaan tuotantoympäristössä toimineessa Kongressinhallintajärjestelmässä.

Sen sijaan tila-avaruuden analysoinnilla ei saatu yhtä hyviä tuloksia. Suurin syy siihen oli se, että tila-avaruuteen perustuvalla analysoinnilla pyritään todistamaan järjestelmän ominaisuuksia. Tässä tapauksessa järjestelmän ominaisuuksia ei kuitenkaan voitu todistaa, sillä mallin ja toteutuksen vastaavuutta ei voitu varmistaa. Lisäksi mallia ja sen alkuasetuksia jouduttiin karsimaan, jotta tila-avaruuden analysointi olisi mahdollista. Mallin tila-avaruus antoi kuitenkin viitteitä suunnitelman oikeellisesta toiminnasta.

Tutkimuksessa mallinnettiin ja analysoitiin Kongressinhallintajärjestelmän abstraktien vastaanotto ja arviointi -prosessia. Tulevaisuudessa voitaisiin mallintaa myös muut Kongressinhallintajärjestelmän osat, jolloin saataisiin varmuus eri osien saumattomasta yhteistoiminnasta. Malli antaisi selkeän kuvan siitä missä järjestyksessä mitkäkin toiminnot tulisi tehdä kongressia hallittaessa. Laajan mallin yhteydessä menetettäisiin mahdollisuus täyden tila-avaruuden analysointiin, mutta mallia voitaisiin edelleen analysoida simuloimalla. Lisäksi jokaisen eri osan oma tila-avaruus olisi mahdollista analysoida.

8 Lähteet

[1] G. Balbo et al.: "An Example of Modelling and Evaluation of a Concurrent Program Using Coloured Stochastic Petri Nets: Lamport's Fast Mutual Exclusion Algorithm", *IEEE Transactions on Parallel and Distributed Systems*, 3(2), ss. 221–240, 1992, julkaistu myös kirjassa [36].

[2] Luca Bernardinello ja Fiorella De Cindio: "A Survey of Basic Net Models and Modular Net Classes", teoksessa "Advances in Petri Nets 1992", osa 609 sarjasta *Lecture Notes in Computer Science*, (ss. 304–351), Springer-Verlag, Berlin, 1992.

[3] "The BETA Programming Language", Viitattu 25.5.2007.
URL `http://www.daimi.au.dk/~beta/`

[4] Barry Boehm: *Software Engineering Economics*, Prentice-Hall, Englewood Cliffs, USA, 1981.

[5] Heino Carstensen ja Rüdiger Valk: "Infinite Behaviour and Fairness in Petri Nets", teoksessa "Advances in Petri Nets 1984", osa 188 sarjasta *Lecture Notes in Computer Science*, (ss. 83–100), Springer-Verlag, Berlin, 1985.

[6] Wai-Kai Chen: *Applied Graph Theory*, osa 13 sarjasta *Applied Mathematics and Mechanics*, North-Holland Publishing Company, Amsterdam, 1971.

[7] Søren Christensen, Lars M. Kristensen ja Thomas Mailund: "A Sweep-Line Method for State Space Exploration", teoksessa "Proceedings of TACAS 2001", osa 2031 sarjasta *Lecture Notes in Computer Science*, (ss. 450–464), Springer-Verlag, Berlin, 2001.

[8] Søren Christensen ja Laure Petrucci: "Modular State Space Analysis of Coloured Petri Nets", teoksessa "Application and Theory of Petri Nets 1995", osa 935 sarjasta *Lecture Notes in Computer Science*, (ss. 201–217), Springer-Verlag, Berlin, 1995.

[9] J.-P. Courtiat, J. M. Ayache ja B. Algayres: "Petri Nets are Good for Protocols", teoksessa "SIGCOMM '84: Proceedings of the ACM SIGCOMM Symposium on

Communications Architectures and Protocols", (ss. 66–74), ACM Press, New York, USA, 1984.

[10] "CPN Group", Viitattu 25.5.2007.
URL http://www.daimi.au.dk/CPnets/

[11] "CPN Tools", Viitattu 25.5.2007.
URL http://wiki.daimi.au.dk/cpntools/cpntools.wiki

[12] "CPN Tools, Known Limitations", Viitattu 25.5.2007.
URL http://wiki.daimi.au.dk/cpntools-help/known_limitations.wiki?cmd=get&anchor=known+limitations

[13] Giovanni Denaro ja Mauro Pezzè: "Petri Nets and Software Engineering", teoksessa "Lectures on Concurrency and Petri Nets", osa 3098 sarjasta *Lecture Notes in Computer Science*, (ss. 439–466), Springer-Verlag, Berlin, 2004.

[14] Jörg Desel ja Wolfgang Reisig: "Place/Transition Petri Nets", teoksessa "Lectures on Petri Nets I: Basic Models", osa 1491 sarjasta *Lecture Notes in Computer Science*, (ss. 122–173), Springer-Verlag, Berlin, 1998.

[15] "Design/CPN", Viitattu 25.5.2007.
URL http://www.daimi.au.dk/designCPN/

[16] "Examples of Industrial Use of CP-nets", Viitattu 25.5.2007.
URL http://www.daimi.au.dk/CPnets/intro/example_indu.html

[17] John B. Fraleigh: *Linear Algebra*, Addison-Wesley, Reading, USA, 1990.

[18] David Garlan ja Mary Shaw: "An Introduction to Software Architecture", teoksessa "Advances in Software Engineering and Knowledge", (ss. 1–39), World Scientific Publishing Company, Singapore, 1993.

[19] Hartmann J. Genrich ja Kurt Lautenbach: "System Modelling with High-level Petri Nets", *Theoretical Computer Science*, 13, ss. 109–136, 1981.

[20] Steven Gordon, Lars M. Kristensen ja Jonathan Billington: "Verification of a Revised WAP Wireless Transaction Protocol", teoksessa "Applications and Theory of Petri Nets 2002", osa 2360 sarjasta *Lecture Notes in Computer Science*, (ss. 182–202), Springer-Verlag, Berlin, 2002.

[21] James Gosling et al.: *The Java Language Specification Third Edition*, Addison-Wesley, 2005.

[22] Serge Haddad: "A Reduction Theory for Coloured Nets", teoksessa "Advances in Petri Nets 1989", osa 424 sarjasta *Lecture Notes in Computer Science*, (ss. 209–235), Springer-Verlag, Berlin, 1990.

[23] Peter Huber, Kurt Jensen ja Robert M. Shapiro: "Hierarchies in Coloured Petri Nets", teoksessa "Advances in Petri Nets 1990", osa 483 sarjasta *Lecture Notes in Computer Science*, (ss. 313–341), Springer-Verlag, Berlin, 1991, julkaistu myös kirjassa [36].

[24] Peter Huber et al.: "Reachability Trees for High-level Petri Nets", *Theoretical Computer Science*, 45, ss. 261–292, 1986, laajennettu versio julkaistu kirjassa [36].

[25] ISO/IEC 15909-1:2004: "High-level Petri Nets–Part 1: Concepts, Definitions and Graphical Notation", 2004.

[26] Matthias Jantzen ja Rüdiger Valk: "Formal Properties of Place/Transition Nets", teoksessa "Net Theory and Applications, Proceedings of the Advanced Course on General Net Theory of Processes and Systems 1979", (ss. 165–212), Springer-Verlag, Berlin, 1980.

[27] Kurt Jensen: "Coloured Petri Nets and the Invariant Method", *Theoretical Computer Science*, 14, ss. 317–336, 1981.

[28] Kurt Jensen: "High-level Petri Nets", teoksessa "Informatik-Fachberichte 66: Application and Theory of Petri Nets", osa 66, (ss. 166–180), Springer-Verlag, Berlin, 1983.

[29] Kurt Jensen: "Coloured Petri Nets", teoksessa "Petri Nets: Central Models and Their Properties", osa 254 sarjasta *Lecture Notes in Computer Science*, (ss. 248–299), Springer-Verlag, Berlin, 1987.

[30] Kurt Jensen: "Coloured Petri Nets: A High Level Language for System Design and Analysis", teoksessa "Advances in Petri Nets 1990", osa 483 sarjasta *Lecture Notes in Computer Science*, (ss. 342–416), Springer-Verlag, Berlin, 1991, julkaistu myös kirjassa [36].

[31] Kurt Jensen: *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use, Volume 2*, Monographs in Theoretical Computer Science, Springer-Verlag, Berlin, 1994.

[32] Kurt Jensen: "An Introduction to the Theoretical Aspects of Coloured Petri Nets", teoksessa "A Decade of Concurrency Reflections and Perspectives", osa 803 sarjasta *Lecture Notes in Computer Science*, (ss. 230–272), Springer-Verlag, Berlin, 1994.

[33] Kurt Jensen: "An Introduction to the Practical Use of Coloured Petri Nets", teoksessa "Lectures on Petri Nets II: Applications", osa 1492 sarjasta *Lecture Notes in Computer Science*, (ss. 237–292), Springer-Verlag, Berlin, 1998.

[34] Kurt Jensen, Søren Christensen ja Lars M. Kristensen: *CPN Tools State Space Manual*, University of Aarhus, Denmark, 2006.
URL `http://wiki.daimi.au.dk/cpntools-help/manual.pdf.wiki?cmd=get&anchor=manual.pdf&type=file`

[35] Kurt Jensen, Lars M. Kristensen ja Lisa Wells: "Coloured Petri Nets and CPN Tools for Modelling and Validation of Concurrent Systems", *International Journal on Software Tools for Technology Transfer*, 9(3-4), ss. 213–254, 2007.

[36] Kurt Jensen ja Grzegorz Rozenberg (toim.): *High-level Petri Nets. Theory and Applications*, Springer-Verlag, Berlin, 1991.

[37] Jens Bæk Jørgensen: "Coloured Petri Nets in UML-Based Software Development. Designing Middleware for Pervasive Healthcare", teoksessa "Proceedings of the Fourth Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools", (ss. 61–80), 2002.

[38] S. Rao Kosaraju: "Decidability of Reachability in Vector Addition Systems", teoksessa "STOC '82: Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing", (ss. 267–281), ACM Press, New York, USA, 1982.

[39] Lars M. Kristensen, Soren Christensen ja Kurt Jensen: "The Practitioner's Guide to Coloured Petri Nets", *International Journal on Software Tools for Technology Transfer*, 2(2), ss. 98–132, 1998.

[40] Lars M. Kristensen ja Laure Petrucci: "An Approach to Distributed State Space Exploration for Coloured Petri Nets", teoksessa "Applications and Theory of

Petri Nets 2004", osa 3099 sarjasta *Lecture Notes in Computer Science*, (ss. 474–483), Springer-Verlag, Berlin, 2004.

[41] Lars M. Kristensen ja Antti Valmari: "Finding Stubborn Sets of Coloured Petri Nets without Unfolding", teoksessa "Application and Theory of Petri Nets 1998", osa 1420 sarjasta *Lecture Notes in Computer Science*, (ss. 104–123), Springer-Verlag, Berlin, 1998.

[42] Charles W. Krueger: "Software Reuse", *ACM Computing Surveys*, 24(2), ss. 131–183, 1992.

[43] Daniel J. Lehmann, Amir Pnueli ja Jonathan Stavi: "Impartiality, Justice and Fairness: The Ethics of Concurrent Termination", teoksessa "Automata, Languages and Programming", osa 115 sarjasta *Lecture Notes in Computer Science*, (ss. 264–277), Springer-Verlag, Berlin, 1981.

[44] David A. Marca ja Clement L. McGowan: *SADT: Structured Analysis and Design Techniques*, McGraw-Hill, New York, USA, 1987.

[45] G. Memmi ja J. Vautherin: "Analysing Nets by the Invariant Method", teoksessa "Petri Nets: Central Models and Their Properties", osa 254 sarjasta *Lecture Notes in Computer Science*, (ss. 300–336), Springer-Verlag, Berlin, 1987, julkaistu myös kirjassa [36].

[46] Robin Milner et al.: *The Definition of Standard ML (Revised)*, MIT Press, Cambridge, USA, 1997.

[47] Tadao Murata: "Circuit Theoretic Analysis and Synthesis of Marked Graphs", *IEEE Transactions on Circuits and Systems*, 24(7), ss. 400–405, 1977.

[48] Tadao Murata: "State Equation, Controllability, and Maximal Matchings of Petri Nets", *IEEE Transactions on Automatic Control*, 22(3), ss. 412–416, 1977.

[49] Tadao Murata: "Petri Nets: Properties, Analysis and Applications", *Proceedings of the IEEE*, 77(4), ss. 541–580, 1989.

[50] Y. Narahari ja N. Viswanadham: "On The Invariants of Coloured Petri Nets", teoksessa "Advances in Petri Nets 1985", osa 222 sarjasta *Lecture Notes in Computer Science*, (ss. 330–345), Springer-Verlag, Berlin, 1986.

[51] Object Management Group (OMG): "Unified Modeling Language Specification, Version 2.1.1", 2007.

[52] "The Perl Directory", Viitattu 25.5.2007.
URL `http://www.perl.org/`

[53] James L. Peterson: "Petri Nets", *ACM Computing Surveys*, 9(3), ss. 223–252, 1977.

[54] James L. Peterson: *Petri Net Theory and the Modeling of Systems*, Prentice-Hall, Englewood Cliffs, USA, 1981.

[55] Carl Adam Petri: *Kommunikation mit Automaten*, Bonn: Institut für Instrumentelle Mathematik, Schriften des IIM Nr. 2, 1962.

[56] Carl Adam Petri: "Communication with Automata", *New York: Griffits Air Force Base, Tekninen raportti RADC-TR-65–377*, 1, ss. 1–Suppl. 1, 1966, englannin kielinen käännös.

[57] Valerio Pinci ja Robert M. Shapiro: "An Integrated Software Development Methodology Based On Hierarchical Colored Petri Nets", teoksessa "Advances in Petri Nets 1991", osa 524 sarjasta *Lecture Notes in Computer Science*, (ss. 227–252), Springer-Verlag, Berlin, 1991, julkaistu myös kirjassa [36].

[58] "PostgreSQL", Viitattu 25.5.2007.
URL `http://www.postgresql.org/`

[59] Roger S. Pressman: *Software Engineering, A Practitioner's Approach*, McGraw-Hill, London, UK, 2000.

[60] Wolfgang Reisig: *Petri Nets, An Introduction*, Springer-Verlag, Berlin, 1985.

[61] Wolfgang Reisig: "Petri Nets in Software Engineering", teoksessa "Petri Nets: Applications and Relationships to Other Models of Concurrency", osa 255 sarjasta *Lecture Notes in Computer Science*, (ss. 62–96), Springer-Verlag, Berlin, 1987.

[62] Wolfgang Reisig: "Place/Transition Systems", teoksessa "Petri Nets: Central Models and Their Properties", osa 254 sarjasta *Lecture Notes in Computer Science*, (ss. 117–141), Springer-Verlag, Berlin, 1987.

[63] Robert M. Shapiro: “Validation of a VLSI Chip Using Hierarchical Colored Petri Nets”, *Microelectronics and Reliability*, 31(4), ss. 607–625, 1991, julkaistu myös kirjassa [36].

[64] M. Silva et al.: “Generalized Inverses and the Calculation of Symbolic Invariants for Coloured Petri Nets”, *Technique et Science Informatiques*, 4(1), ss. 113–126, 1985, julkaistu myös kirjassa [36].

[65] “Standard ML of New Jersey”, Viitattu 25.5.2007.
URL `http://www.smlnj.org/`

[66] Antti Valmari: “Stubborn Sets for Reduced State Space Generation”, teoksessa “Advances in Petri Nets 1990”, osa 483 sarjasta *Lecture Notes in Computer Science*, (ss. 491–515), Springer-Verlag, Berlin, 1991.

[67] Antti Valmari: “Compositional State Space Generation”, teoksessa “Advances in Petri Nets 1993”, osa 674 sarjasta *Lecture Notes in Computer Science*, (ss. 427–457), Springer-Verlag, Berlin, 1993.

[68] Antti Valmari: “The State Explosion Problem”, teoksessa “Lectures on Petri Nets I: Basic Models”, osa 1491 sarjasta *Lecture Notes in Computer Science*, (ss. 429–528), Springer-Verlag, Berlin, 1998.

A Kongressinhallintajärjestelmän mallinnuksessa käytetyt määritelmät

Liitteessä esitetään CPN Tools -sovelluksessa käytetyt määritelmät Kongressinhallintajärjestelmän mallintamisessa. Määrittelyt ovat eroteltuna värijoukkojen, muuttujien, vakioiden, paikkojen alustuksien ja funtioiden määrittelyihin.

A.1 Värijoukkojen määrittelyt

Liite sisältää määritetyt värijoukot. Tietueiden ja monikoiden yhteydessä on kuvattu mitä eri kentät tarkoittavat.

```
colset UNIT = unit;
colset INT = int;
colset BOOL = bool;
colset STRING = string;
colset ID = INT;
colset TOPIC = INT;
colset TOPICS = list ID;
colset PROBABILITY = int with 0..4;
colset SCORE = int with 1..4;
colset DEFAULT = BOOL;
colset INVITED = BOOL;
```

Värijoukko `DECISION` sisältää järjestäjän abstraktille tekemät eri päätökset. *Acc* tarkoittaa abstraktin hyväksymistä, *Rej* abstraktin hylkäämistä, *Mod* abstraktin lähettämistä muokattavaksi, ja *NewRev* tarkoittaa uuden arvioijan etsimistä abstraktille.

```
colset DECISION = with Acc | Rej | Mod | NewRev;
colset REVID = ID;
```

Värijoukko `REV_LAST` sisältää arvioijan ID-numeron sekä totuusarvon. Liitettynä aihealueeseen, totuusarvo kuvaa onko kyseinen arvioija valittu viimeksi kyseiseen aihealueeseen asetetun abstraktin arvioijaksi.

```
colset REV_LAST = product REVID * BOOL;
colset REVS = list REV_LAST;
```

Värijoukko TOP_REVS sisältää parin, jossa on aihealueen ID-numero sekä lista aihealueen arvioijien REV_LAST pareista.

```
colset TOP_REVS = product TOPIC * REVS;
colset TOPS_REVS = list TOP_REVS;
```

Värijoukko USER sisältää järjestelmän käyttäjän identifioimiseen tarvittavat tiedot. Itse mallinnuksessa muulla kuin ID-numerolla ei ole merkitystä.

```
colset USER = record id:ID *
                     first_name:STRING *
                     last_name:STRING *
                     email:STRING;
colset ORGANIZER = USER;
```

Värijoukko SESSIONTYPE sisältää kolmikon, jossa on sessiotyypin ID, totuusarvo, joka kuvaa onko sessiotyyppi kutsuttuja abstrakteja varten, ja totuusarvo, joka kuvaa onko kyseinen sessiotyyppi oletussessiotyyppi.

```
colset SESSIONTYPE = product ID * INVITED * DEFAULT;
colset SESSIONTYPES = list SESSIONTYPE;
```

Värijoukko SESSION sisältää session ID-numeron, aihealueen sekä sessiotyypin.

```
colset SESSION = record id:ID *
                        top:TOPIC *
                        stype:SESSIONTYPE;
colset SESSIONS = list ID;
```

Värijoukko SES_ORGANIZER sisältää session järjestäjän henkilökohtaiset tiedot sekä listan sessioiden ID-numeroista, joita kyseinen session järjestäjä on järjestämässä.

```
colset SES_ORGANIZER = record org:USER *
                              sessions:SESSIONS;
```

Värijoukko `ORG_SES` sisältää parin, jossa on järjestäjän ID sekä session ID, johon järjestäjä on kutsumassa uutta abstraktia.

```
colset ORG_SES = product ID * ID;
```

Värijoukko `SCASE` sisältää parin, jossa on erikoistapauksen ID-numero sekä erikoistapaukselle määritellyn arvioijan ID-numero. Mikäli erikoistapaukselle ei ole asetettu arvioijaa, on arvioijan ID `0`.

```
colset SCASE = product ID * REVID;
```

Värijoukko `SPECIALCASE` on yhdiste värijoukoista `SCASE` ja `UNIT`. Värijoukko sisältää siis erikoistapauksen tai tyhjän arvon.

```
colset SPECIALCASE = union SC:SCASE + NSC:UNIT;
```

Värijoukko `ABS_STYPE` on yhdiste värijoukoista `SESSIONTYPE` ja `UNIT`. Tällöin voidaan valita joko jokin sessiotyyppi tai tyhjä arvo.

```
colset ABS_STYPE = union ST:SESSIONTYPE + NST:UNIT;
```

Värijoukko `ABS_SES` on yhdiste värijoukoista `ID` ja `UNIT`. Sen avulla voidaan kuvata ID-viitettä tai asettaa viitteeksi tyhjä arvo.

```
colset ABS_SES = union SES:ID + NS:UNIT;
```

Värijoukko `ABSTRACT` sisältää abstraktin tiedot. Se on tietue, joka sisältää abstraktin ID-numeron sekä aihealueen ID-numeron. Abstraktin kenttä `stype` viittaa joko sessiotyyppiin tai se voi olla tyhjä arvo. Samaan tapaan kenttä `session` viittaa joko sessioon, johon abstrakti kuuluu, tai tyhjään arvoon, mikäli abstrakti ei kuulu sessioon. Kenttä `cp` sisältää yhteyshenkilön ID-numeron. Kentässä `sc` säilytetään abstraktiin liitettyä erikoistapausta tai sitten se voi sisältää tyhjän arvon. Kongressinhallintajärjestelmässä abstrakti voi olla liitettynä useaan erikoistapaukseen, mutta mallinnuksessa tyydyttiin mallintamaan vain yhtä erikoistapausta. Lisäksi Kongressinhallintajärjestelmän aikana voidaan erikoistapaukseen liitettyä arvioijaa vaihtaa. Abstraktiin voidaan kuitenkin liittää erikoistapaus arvioijineen, sillä mallissa erikoistapauksen arvioijaa ei voida muuttaa.

```
colset ABSTRACT = record id:ID *
                         top:TOPIC *
                         stype:ABS_STYPE *
                         session:ABS_SES *
                         cp:ID *
                         sc:SPECIALCASE;
```

Värijoukko ABSREV sisältää kolmikon, jossa on abstraktin ID, kyseisen abstraktin arvioijan ID sekä totuusarvo siitä onko arvioija jo arvioinut abstraktin.

```
colset ABSREV = product ID * ID * BOOL;
colset ABSREVS = list ABSREV;
```

Värijoukko SETTINGS sisältää arviointiin liittyvät Kongressinhallintajärjestelmän asetukset. Kenttä reviewer_select sisältää tiedon siitä onko arvioijien valinta päällä vai meneekö jokainen abstrakti suoraan järjestäjille arvioitavaksi. Kenttä accepted sisältää kokonaislukuarvon, mistä lähtien abstraktit hyväksytään suoraan arvioijan arvosanan perusteella.

```
colset SETTINGS = record reviewer_select: BOOL *
                         accepted: INT;
```

A.2 Käytettyjen muuttujien määrittelyt

Liite sisältää määritetyt muuttujat.

```
var count: INT;
var absid: ID;
var id: ID;
var orgid: ID;
var sesid: ID;
var userid: ID;
var abs: ABSTRACT;
var absrevs: ABSREVS;
var d: DECISION;
var not_rev: BOOL;
var org: ORGANIZER;
```

```
var s: PROBABILITY;
var sc: SPECIALCASE;
var score: SCORE;
var ses: SESSION;
var ses_org: SES_ORGANIZER;
var set: SETTINGS;
var stypes: SESSIONTYPES;
var top: TOPIC;
var trs: TOPS_REVS;
var usr: USER;
```

A.3 Määritellyt vakiot

Liite sisältää mallinnuksessa käytetyt vakioarvot.

```
val d0 = 0;
val d1 = 1;
val d3 = 1;
val d4 = 0;
val d5 = 0;
val max_abstracts = 1;
```

A.4 Paikkojen alustukset

Liitteessä esitellään paikoille määritetyt alkumerkkaukset.

```
val init_abstractid = 1000;
val init_contacts = {id=6,first_name="",
                     last_name="",email=""}
val init_organizers = {id=1,first_name="",
                       last_name="",email=""}
val init_ses_orgs = {org={id=2,first_name="",
                          last_name="",email=""},
                     sessions=[1,3]}
val init_sessions = 1‘{id=1,top=1,stype=(1,false,true)}++
                    1‘{id=2,top=2,stype=(2,true,false)}++
```

```
                    1‘{id=3,top=3,stype=(3,false,false)}
val init_settings = {reviewer_select = true,accepted = 3}
val init_specialcases = 1‘SC(1,4)++1‘SC(2,0)++1‘NSC()
val init_stypes = [(1,false,true),
                   (2,true,false),
                   (3,false,false)]
val init_topics = 1‘1++1‘2++1‘3++1‘4
val init_tops_revs = [(1,[(3,false),(4,false)]),
                      (2,[(3,false),(5,false)]),
                      (3,[(5,false)])]
```

A.5 Funktiot

Liite sisältää mallin suorituksen aikana käytetyt funktiot. Ennen funktion määrittelyä on annettu lyhyt kuvaus sen toiminnasta.

Funktio `invited` tutkii kuuluuko parametrina annettu abstrakti kutsutun tyyppiseen sessiotyyppiin vai ei, ja palauttaa totuusarvon.

```
fun invited (abs:ABSTRACT) =
  case #stype abs of
    ST(_,true,_) => true
  | _            => false
```

Funktio `check_top_rev` tutkii onko parametrina annetulle abstraktin aihealueelle määritelty arvioijia. Toisena parametrina on aihealue ja lista kyseisen aihealueen arvioijista. Funktio palauttaa totuusarvon.

```
fun check_top_rev (abs:ABSTRACT) (tr:TOP_REVS) =
  #top abs = (#1 tr) andalso not (List.null (#2 tr));
```

Funktio `next_top_rev` etsii aihealueen seuraavan arvioijan parametrina annetusta aihealueen arvioijien listasta. Lista sisältää pareja, joissa on arvioijan ID-numero sekä totuusarvo. Totuusarvo kuvaa arvioijaa, joka on viimeksi valittu aihealueeseen kuuluvan abstraktin arvioijaksi. Näin ollen seuraavaksi valitaan listassa tätä seuraava arvioija tai ensimmäinen mikäli viimeksi valittu arvioija oli listan viimeinen. Mikäli lista ei sisällä yhtään arvioijaa, palautetaan 0.

```
fun next_top_rev (rev::revs:REVS) false 0 =
  (case #2 rev of
      true  => next_top_rev revs true (#1 rev)
    | false => next_top_rev revs false (#1 rev))
| next_top_rev (rev::revs:REVS) false first =
  (case #2 rev of
      true  => next_top_rev revs true first
    | false => next_top_rev revs false first)
| next_top_rev (rev::revs) true _ = #1 rev
| next_top_rev nil _ 0 = 0
| next_top_rev nil _ first = first
```

Funktio `assigned_rev` tutkii onko abstraktille määritetty arvioija `ABSREVS` tyyppisessä arvossa. Mikäli arvioija on määritetty, funktio palauttaa kyseisen tyyppisen kolmikon arvioinnin arvoksi asetettuna `false`. Muussa tapauksessa funktio palauttaa tyhjän monijoukon.

```
fun assigned_rev (abs:ABSTRACT) (a::ar:ABSREVS) =
  (case #id abs = #1 a of
     true  => 1‘(#1 a,#2 a,false)
   | false => assigned_rev abs ar)
| assigned_rev abs nil = empty
```

Funktio `find_top_rev` etsii abstraktille aihealueen seuraavan arvioijan. Mikäli aihealueelle ei ole määritelty arvioijia, funktio palauttaa tyhjän monijoukon.

```
fun find_top_rev abs tr =
  case check_top_rev abs tr of
     true  => 1‘(#id abs,next_top_rev (#2 tr) false 0,false)
   | false => empty
```

Funktio `select_reviewer` etsii abstraktille arvioijaa. Mikäli abstrakti kuuluu erikoistapaukseen, jolle on määrätty arvioija, valitaan kyseinen arvioija abstraktille arvioijaksi. Muussa tapauksessa tutkitaan onko abstraktille jo asetettu arvioija ja valitaan se. Mikäli abstraktille ei ole asetettu arvioijaa, etsitään abstraktin aihealueeseen määrättyjen arvioijien joukosta seuraavaa. Funktio palauttaa tyhjän monijoukon mikäli arvioijaa ei löydy. Muussa tapauksessa palauttaa monijoukon, joka

sisältää yhden kolmikon, jossa on abstraktin ID, arvioijan ID ja totuusarvo `false`, joka kuvaa, että abstraktia ei ole arvioitu.

```
fun select_reviewer abs (tr::trs) ar =
  (let
    val arev = assigned_rev abs ar
    val toparev = find_top_rev abs tr
   in
    (case #sc abs of
       NSC()   => (case size arev of
                     1 => arev
                   | _ => (case size toparev of
                             1  => toparev
                           | _  => select_reviewer abs trs ar))
     | SC(_,0) => (case size arev of
                     1  => arev
                   | _  => (case size toparev of
                              1 => toparev
                            | _ => select_reviewer abs trs ar))
     | SC(_,revid) =>  1‘(#id abs,revid,false))
   end)
| select_reviewer abs nil ar = assigned_rev abs ar
```

Funktio `in_session` palauttaa totuusarvon siitä kuuluuko parametrina annettu abstrakti sessioon vai ei.

```
fun in_session (abs:ABSTRACT) =
  case #session abs of
    SES(_) => true
  | NS()   => false
```

Funktio `to_organizer` tutkii tuleeko abstrakti lähettää järjestäjille vai arvioijille arvioitavaksi. Mikäli abstrakti kuuluu kutsuttuun sessiotyyppiin, sessioon, tai arvioijien valitseminen on poissa käytöstä, abstrakti lähetetään järjestäjille. Muussa tapauksessa etsitään arvioijaa. Mikäli arvioijaa ei löydy, lähetetään abstrakti järjestäjille, muussa tapauksessa arvioijille. Funktio palauttaa totuusarvon.

```
fun to_organizer abs trs (set:SETTINGS) =
  case invited abs orelse not (#reviewer_select set)
       orelse in_session abs of
    true  => true
  | false => case size (select_reviewer abs trs []) of
               1 => false
             | _ => true;
```

Funktiota `new_decision` käytetään luomaan satunnaisuutta järjestäjän tehdessä uutta päätöstä. Funktio ottaa parametrina abstraktin ja kaksi vertailtavissa olevaa arvoa. Mikäli ensimmäinen arvo on pienempi kuin toinen, funktio palauttaa monijoukon, joka sisältää abstraktin. Muussa tapauksessa funktio palauttaa tyhjän monijoukon.

```
fun new_decision abs d s =
  case s < d of
    true  => 1‘abs
  | false => empty
```

Funktio `no_new_decision` toimii käänteisesti verrattuna funktioon `new_decision`.

```
fun no_new_decision abs d s =
  case s >= d of
    true  => 1‘abs
  | false => empty
```

Funktio `update_rev_abs` päivittää paikassa *Abs_Rev* olevaa listaa valitun arvioijan mukaan. Mikäli lista sisältää kolmikon, jossa on abstraktin ja tämän arvioijan ID-numerot, päivitetään kolmikon kolmannen kentän totuusarvoksi `false`. Mikäli listassa on abstraktille määriteltynä toinen arvioija, jätetään tämä alkio pois uudesta listasta. Lopuksi lisätään uusi kolmikko listan loppuun mikäli listasta ei löytynyt kolmikkoa, joka sisältäisi sekä abstraktin että sen arvioijan ID-numerot.

```
fun update_rev_abs abs new (ar::ars) =
  (case #1 new = #1 ar of
     true  => (case #2 new = #2 ar of
                 true  => (#1 new,#2 new,false)::ars
```

```
                | false => update_rev_abs abs new ars)
   | false => ar::update_rev_abs abs new ars)
| update_rev_abs abs new nil = [new];
```

Funktio `update_absrevs` päivittää paikassa *Abs_Rev* olevaa listaa mikäli abstraktille löytyy arvioija. Kutsuu päivitykseen funktiota `update_rev_abs`.

```
fun update_absrevs abs trs ar =
  let
    val rev = select_reviewer abs trs ar
  in
    case size rev of
      1 => update_rev_abs abs (ms_to_col(rev)) ar
    | _ => ar
  end;
```

Funktio `update_top_revs` päivittää abstraktin aihealueeseen liittyvää listaa paikassa *Topics Reviewers*. Funktio etsii abstraktille valitun arvioijan, ja päivittää listaan tämän arvioijan valituksi asettamalla kyseisen arvioijan `BOOL`-kentän arvoksi `true`. Muiden arvioijien `BOOL`-kentän arvoksi asetetaan `false`.

```
fun update_top_revs tr abs ar =
  let
    val rev = ms_to_col (select_reviewer abs [tr] ar)
  in
    (#1 tr,map (fn x => if (#2 rev = #1 x)
                        then (#1 x,true)
                        else (#1 x,false)) (#2 tr))
  end;
```

Funktio `update_tops_revs` päivittää paikassa *Topics Reviewers* olevaa listaa. Funktio etsii listasta oikean aihealueen, ja päivittää aihealueeseen asetettujen arvioijien listaa funktion `update_top_revs` avulla.

```
fun update_tops_revs (tr::trs) abs ar =
  (case #1 tr = #top abs of
        true => update_top_revs tr abs ar::trs
     | false => tr::update_tops_revs trs abs ar)
| update_tops_revs nil _ _ = [];
```

Funktio `default_stype` etsii sessiotyyppien joukosta oletussessiotyyppiä, ja palauttaa tämän mikäli sellainen löytyy. Mikäli oletussessiotyyppiä ei ole asetettu, funktio palauttaa `UNIT`-tyyppisen arvon.

```
fun default_stype (st::sts) =
  (case (#3 st) of
     true  => ST(st)
  |  false => default_stype sts)
| default_stype nil = NST()
```

Funktio `reviewed` päivittää paikan *Abs_Rev* listaan abstraktin arvioiduksi.

```
fun reviewed id (ar::ars) =
  (case id = #1 ar of
     true  => (#1 ar,#2 ar,true)::ars
   | false => ar::reviewed id ars)
| reviewed id nil = []
```

Funktio `check_mandatory` tutkii, että kaikki abstraktin pakolliset tiedot on annettu ennen arviointiin jättämistä. Mallin tapauksessa riittää tutkia, että abstraktille on asetettu sessiotyyppi.

```
fun check_mandatory (abs:ABSTRACT) =
  case #stype abs of
    ST(_) => true
  | NST() => false
```

Funktio `has_reviewer` tutkii, että abstraktille on asetettu arvioija. Funktio palauttaa totuusarvon.

```
fun has_reviewer (abs:ABSTRACT) (a::ar:ABSREVS) =
  (case #id abs = #1 a of
     true  => true
   | false => has_reviewer abs ar)
| has_reviewer abs nil = false;
```

Funktio `to_sc_rev` tutkii lähetetäänkö parametrina annettu abstrakti arviointiin erikoistapauksen arvioijalle. Mikäli abstraktille ei ole asetettu erikoistapausta,

tai abstraktin erikoistapaukselle ei ole asetettu arvioijaa, funktio palauttaa arvon `false`. Funktio palauttaa arvon `true` mikäli abstrakti on asetettu erikoistapaukseen, jolle on asetettu arvioija.

```
fun to_sc_rev (abs:ABSTRACT) =
  case #sc abs of
    NSC()   => false
  | SC(_,0) => false
  | _       => true
```

Funktio `new_rev` simuloi järjestäjän abstraktille valitsemaa uutta arvioijaa. Funktio ottaa abstraktin aihealueen arvioijista satunnaisen ja palauttaa kolmikon, jossa on abstraktin ID, arvioijan ID sekä totuusarvo `false`. Funktio palauttaa tyhjän monijoukon mikäli aihealueelle ei ole olemassa arvioijia.

```
fun new_rev (abs:ABSTRACT) (tr::trs:TOPS_REVS) =
  (case #top abs = #1 tr of
     true  => (case List.null (#2 tr) of
                 true  => empty
               | false =>
                   let
                     val n = discrete(0,length(#2 tr)-1)
                   in
                     1‘(#id abs,#1 (List.nth(#2 tr,n)),false)
                   end)
   | false => new_rev abs trs)
| new_rev abs nil = empty
```

B Kongressinhallintajärjestelmän mallin tila-avaruuden raportti

Liitteessä esitetään CPN Tools -sovelluksen tuottama mallin tila-avaruuden raportti. Raportti sisältää yleistietoa mallin tila-avaruudesta sekä mallista analysoidut rajoittuvuus-, kotitila-, elävyys- ja reiluusominaisuudet.

B.1 Tilastotietoa mallin tila-avaruudesta

State Space	
Nodes:	29094
Arcs:	849074
Secs:	8025
Status:	Full

Scc Graph	
Nodes:	374
Arcs:	13642
Secs:	81

B.2 Mallista analysoidut rajoittuvuusominaisuudet

Liitteessä esitetään paikkojen merkkien lukumäärien ylä- ja alarajat. Tämän jälkeen esitetään paikkojen monijoukkojen ylä- ja alarajat, jotka on luettavuuden takia erotettu toisistaan.

B.2.1 Parhaat paikkojen merkkien lukumäärien rajat

Best Integer Bounds		
Page'Place	Upper	Lower
Abstract_Process'Accepted 1	1	0
Abstract_Process'Failed 1	1	0
Abstract_Process'Modify 1	1	0
Abstract_Process'Organizer 1	1	0
Abstract_Process'Saved 1	1	0
Abstract_Process'Submitted 1	1	0
Contact_Edit_Abs'Contact_Person 1	1	1
Contact_Edit_Abs'Session_types 1	1	1
Contact_Edit_Abs'Special_Cases 1	3	3
Contact_Modify_Abs'Contact_Person 1	1	1
Decide'Find_Reviewer 1	1	0
Decide'Wait_Decision 1	1	0
Invite_Abs'Abstract_Ids 1	1	1
Invite_Abs'Contact_Person 1	1	1
Invite_Abs'Invited_By_In 1	1	0
Invite_Abs'Sessions 1	3	3
Invite_By_Org'Abs_Count 1	1	1
Invite_By_Org'Organizers 1	1	1
Invite_By_Ses_Org'Abs_Count 1	1	1
Invite_By_Ses_Org'Session_Organizers 1	1	1
Make_Decision'Organizers 1	1	1
Org_Edit_Abs'Organizers 1	1	1
Org_Edit_Abs'Organizers 2	1	1
Org_Edit_Abs'Organizers 3	1	1
Org_Edit_Abs'Organizers 4	1	1
Org_Edit_Abs'Organizers 5	1	1
Org_Edit_Abs'Organizers 6	1	1
Org_Edit_Abs'Session_types 1	1	1
Org_Edit_Abs'Session_types 2	1	1
Org_Edit_Abs'Session_types 3	1	1
Org_Edit_Abs'Session_types 4	1	1

Page'Place	Upper	Lower
Org_Edit_Abs'Session_types 5	1	1
Org_Edit_Abs'Session_types 6	1	1
Org_Edit_Abs'Special_Cases 1	3	3
Org_Edit_Abs'Special_Cases 2	3	3
Org_Edit_Abs'Special_Cases 3	3	3
Org_Edit_Abs'Special_Cases 4	3	3
Org_Edit_Abs'Special_Cases 5	3	3
Org_Edit_Abs'Special_Cases 6	3	3
Org_Edit_Abs'Topics 1	4	4
Org_Edit_Abs'Topics 2	4	4
Org_Edit_Abs'Topics 3	4	4
Org_Edit_Abs'Topics 4	4	4
Org_Edit_Abs'Topics 5	4	4
Org_Edit_Abs'Topics 6	4	4
RequestID'Abs_Count 1	1	1
RequestID'Abstract_Ids 1	1	1
RequestID'Contact_Person 1	1	1
RequestID'Session_types 1	1	1
RequestID'Topics 1	4	4
Review'Abs_Rev 1	1	1
Review'Settings 1	1	1
Select_New_Reviewer'Abs_Rev 1	1	1
Select_New_Reviewer'Reviewers 1	1	1
Submit'Abs_Rev 1	1	1
Submit'Abs_Rev 2	1	1
Submit'Settings 1	1	1
Submit'Settings 2	1	1
Submit'Topics_Reviewers 1	1	1
Submit'Topics_Reviewers 2	1	1

B.2.2 Parhaat paikkojen monijoukkojen ylärajat

Best Upper Multi-set Bounds:

Abstract_Process'Accepted 1:

Olkoon $T = \{1, 2, 3, 4\}$, $ST = \{ST((1, F, T)), ST((2, T, F)), ST((3, F, F))\}$,

$SES = \{SES(1), SES(2), SES(3), NS(())\}$ ja $SC = \{SC((1,4)), SC((2,0)), NSC(())\}$. Tällöin paikan monijoukon yläraja saa 144 alkioisen monijoukon: $\sum_{t \in T, st \in ST, ses \in SES, sc \in SC}(1'\{\text{id} = 1000, \text{top} = t, \text{stype} = st, \text{session} = ses, \text{cp} = 6, \text{sc} = sc\})$

Abstract_Process'Failed 1: Sama kuin paikalla Abstract_Process'Accepted 1
Abstract_Process'Modify 1: Sama kuin paikalla Abstract_Process'Accepted 1
Abstract_Process'Organizer 1: Sama kuin paikalla Abstract_Process'Accepted 1
Abstract_Process'Saved 1: Sama kuin paikalla Abstract_Process'Accepted 1
Abstract_Process'Submitted 1: Sama kuin paikalla Abstract_Process'Accepted 1
Contact_Edit_Abs'Contact_Person 1: 1'{id=6,first_name="",last_name="",email=""}
Contact_Edit_Abs'Session_types 1: 1'[(1,false,true),(2,true,false),(3,false,false)]
Contact_Edit_Abs'Special_Cases 1: 1'SC((1,4))++1'SC((2,0))++1'NSC(())
Contact_Modify_Abs'Contact_Person 1:
1'{id=6,first_name="",last_name="",email=""}
Decide'Find_Reviewer 1: Sama kuin paikalla Abstract_Process'Accepted 1
Decide'Wait_Decision 1: Sama kuin paikalla Abstract_Process'Accepted 1
Invite_Abs'Abstract_Ids 1: 1'1000++1'1001
Invite_Abs'Contact_Person 1: 1'{id=6,first_name="",last_name="",email=""}
Invite_Abs'Invited_By_In 1: 1'(1,1)++1'(1,2)++1'(1,3)++1'(2,1)++1'(2,3)
Invite_Abs'Sessions 1:
1'{id=1,top=1,stype=(1,false,true)}++1'{id=2,top=2,stype=(2,true,false)}++
1'{id=3,top=3,stype=(3,false,false)}
Invite_By_Org'Abs_Count 1: 1'0++1'1
Invite_By_Org'Organizers 1: 1'{id=1,first_name="",last_name="",email=""}
Invite_By_Ses_Org'Abs_Count 1: 1'0++1'1
Invite_By_Ses_Org'Session_Organizers 1:
1'{org={id=2,first_name="",last_name="",email=""},sessions=[1,3]}
Make_Decision'Organizers 1: 1'{id=1,first_name="",last_name="",email=""}
Org_Edit_Abs'Organizers 2: 1'{id=1,first_name="",last_name="",email=""}
Org_Edit_Abs'Organizers 3: 1'{id=1,first_name="",last_name="",email=""}
Org_Edit_Abs'Organizers 4: 1'{id=1,first_name="",last_name="",email=""}
Org_Edit_Abs'Organizers 5: 1'{id=1,first_name="",last_name="",email=""}
Org_Edit_Abs'Organizers 6: 1'{id=1,first_name="",last_name="",email=""}
Org_Edit_Abs'Session_types 1: 1'[(1,false,true),(2,true,false),(3,false,false)]
Org_Edit_Abs'Session_types 2: 1'[(1,false,true),(2,true,false),(3,false,false)]

Org_Edit_Abs'Session_types 3: 1'[(1,false,true),(2,true,false),(3,false,false)]
Org_Edit_Abs'Session_types 4: 1'[(1,false,true),(2,true,false),(3,false,false)]
Org_Edit_Abs'Session_types 5: 1'[(1,false,true),(2,true,false),(3,false,false)]
Org_Edit_Abs'Session_types 6: 1'[(1,false,true),(2,true,false),(3,false,false)]
Org_Edit_Abs'Special_Cases 1: 1'SC((1,4))++1'SC((2,0))++1'NSC(())
Org_Edit_Abs'Special_Cases 2: 1'SC((1,4))++1'SC((2,0))++1'NSC(())
Org_Edit_Abs'Special_Cases 3: 1'SC((1,4))++1'SC((2,0))++1'NSC(())
Org_Edit_Abs'Special_Cases 4: 1'SC((1,4))++1'SC((2,0))++1'NSC(())
Org_Edit_Abs'Special_Cases 5: 1'SC((1,4))++1'SC((2,0))++1'NSC(())
Org_Edit_Abs'Special_Cases 6: 1'SC((1,4))++1'SC((2,0))++1'NSC(())
Org_Edit_Abs'Topics 1: 1'1++1'2++1'3++1'4
Org_Edit_Abs'Topics 2: 1'1++1'2++1'3++1'4
Org_Edit_Abs'Topics 3: 1'1++1'2++1'3++1'4
Org_Edit_Abs'Topics 4: 1'1++1'2++1'3++1'4
Org_Edit_Abs'Topics 5: 1'1++1'2++1'3++1'4
Org_Edit_Abs'Topics 6: 1'1++1'2++1'3++1'4
RequestID'Abs_Count 1: 1'0++1'1
RequestID'Abstract_Ids 1: 1'1000++1'1001
RequestID'Contact_Person 1: 1'{id=6,first_name="",last_name="",email=""}
RequestID'Session_types 1: 1'[(1,false,true),(2,true,false),(3,false,false)]
RequestID'Topics 1: 1'1++1'2++1'3++1'4
Review'Abs_Rev 1:
    1'[]++1'[(1000,3,false)]++1'[(1000,3,true)]++1'[(1000,4,false)]++
    1'[(1000,4,true)]++1'[(1000,5,false)]++1'[(1000,5,true)]
Review'Settings 1: 1'{reviewer_select=true,accepted=3}
Select_New_Reviewer'Abs_Rev 1:
    1'[]++1'[(1000,3,false)]++1'[(1000,3,true)]++1'[(1000,4,false)]++
    1'[(1000,4,true)]++1'[(1000,5,false)]++1'[(1000,5,true)]
Select_New_Reviewer'Reviewers 1:
    1'[(1,[(3,false),(4,false)]),(2,[(3,false),(5,false)]),(3,[(5,false)])]++
    1'[(1,[(3,false),(4,false)]),(2,[(3,false),(5,false)]),(3,[(5,true)])]++
    1'[(1,[(3,false),(4,false)]),(2,[(3,false),(5,true)]),(3,[(5,false)])]++
    1'[(1,[(3,false),(4,false)]),(2,[(3,false),(5,true)]),(3,[(5,true)])]++
    1'[(1,[(3,false),(4,false)]),(2,[(3,true),(5,false)]),(3,[(5,false)])]++
    1'[(1,[(3,false),(4,false)]),(2,[(3,true),(5,false)]),(3,[(5,true)])]++

1'[(1,[(3,false),(4,true)]),(2,[(3,false),(5,false)]),(3,[(5,false)])]++
1'[(1,[(3,false),(4,true)]),(2,[(3,false),(5,false)]),(3,[(5,true)])]++
1'[(1,[(3,false),(4,true)]),(2,[(3,false),(5,true)]),(3,[(5,false)])]++
1'[(1,[(3,false),(4,true)]),(2,[(3,false),(5,true)]),(3,[(5,true)])]++
1'[(1,[(3,false),(4,true)]),(2,[(3,true),(5,false)]),(3,[(5,false)])]++
1'[(1,[(3,false),(4,true)]),(2,[(3,true),(5,false)]),(3,[(5,true)])]++
1'[(1,[(3,true),(4,false)]),(2,[(3,false),(5,false)]),(3,[(5,false)])]++
1'[(1,[(3,true),(4,false)]),(2,[(3,false),(5,false)]),(3,[(5,true)])]++
1'[(1,[(3,true),(4,false)]),(2,[(3,false),(5,true)]),(3,[(5,false)])]++
1'[(1,[(3,true),(4,false)]),(2,[(3,false),(5,true)]),(3,[(5,true)])]++
1'[(1,[(3,true),(4,false)]),(2,[(3,true),(5,false)]),(3,[(5,false)])]++
1'[(1,[(3,true),(4,false)]),(2,[(3,true),(5,false)]),(3,[(5,true)])]

Submit'Abs_Rev 1:
1'[]++1'[(1000,3,false)]++1'[(1000,3,true)]++1'[(1000,4,false)]++
1'[(1000,4,true)]++1'[(1000,5,false)]++1'[(1000,5,true)]

Submit'Abs_Rev 2:
1'[]++1'[(1000,3,false)]++1'[(1000,3,true)]++1'[(1000,4,false)]++
1'[(1000,4,true)]++1'[(1000,5,false)]++1'[(1000,5,true)]

Submit'Settings 1: 1'{reviewer_select=true,accepted=3}

Submit'Settings 2: 1'{reviewer_select=true,accepted=3}

Submit'Topics_Reviewers 1: Sama kuin paikalla Select_New_Reviewer'Reviewers 1

Submit'Topics_Reviewers 2: Sama kuin paikalla Select_New_Reviewer'Reviewers 1

B.2.3 Parhaat paikkojen monijoukkojen alarajat

Best Lower Multi-set Bounds:

Abstract_Process'Accepted 1: empty

Abstract_Process'Failed 1: empty

Abstract_Process'Modify 1: empty

Abstract_Process'Organizer 1: empty

Abstract_Process'Saved 1: empty

Abstract_Process'Submitted 1: empty

Contact_Edit_Abs'Contact_Person 1:
1'{id=6, first_name="", last_name="", email=""}

Contact_Edit_Abs'Session_types 1: 1'[(1,false,true), (2,true,false), (3,false,false)]

Contact_Edit_Abs'Special_Cases 1: 1'SC((1,4))++1'SC((2,0))++1'NSC(())

Contact_Modify_Abs'Contact_Person 1:
1'{id=6, first_name="", last_name="", email=""}
Decide'Find_Reviewer 1: empty
Decide'Wait_Decision 1: empty
Invite_Abs'Abstract_Ids 1: empty
Invite_Abs'Contact_Person 1: 1'{id=6, first_name="", last_name="", email=""}
Invite_Abs'Invited_By_In 1: empty
Invite_Abs'Sessions 1:
1'{id=1,top=1,stype=(1,false,true)}++1'{id=2,top=2,stype=(2,true,false)}++
1'{id=3,top=3,stype=(3,false,false)}
Invite_By_Org'Abs_Count 1: empty
Invite_By_Org'Organizers 1: 1'{id=1, first_name="", last_name="", email=""}
Invite_By_Ses_Org'Abs_Count 1: empty
Invite_By_Ses_Org'Session_Organizers 1:
1'{org={id=2,first_name="",last_name="", email=""}, sessions=[1,3]}
Make_Decision'Organizers 1: 1'{id=1, first_name="", last_name="", email=""}
Org_Edit_Abs'Organizers 1: 1'{id=1, first_name="", last_name="", email=""}
Org_Edit_Abs'Organizers 2: 1'{id=1, first_name="", last_name="", email=""}
Org_Edit_Abs'Organizers 3: 1'{id=1, first_name="", last_name="", email=""}
Org_Edit_Abs'Organizers 4: 1'{id=1, first_name="", last_name="", email=""}
Org_Edit_Abs'Organizers 5: 1'{id=1, first_name="", last_name="", email=""}
Org_Edit_Abs'Organizers 6: 1'{id=1, first_name="", last_name="", email=""}
Org_Edit_Abs'Session_types 1: 1'[(1,false,true), (2,true,false), (3,false,false)]
Org_Edit_Abs'Session_types 2: 1'[(1,false,true), (2,true,false), (3,false,false)]
Org_Edit_Abs'Session_types 3: 1'[(1,false,true), (2,true,false), (3,false,false)]
Org_Edit_Abs'Session_types 4: 1'[(1,false,true), (2,true,false), (3,false,false)]
Org_Edit_Abs'Session_types 5: 1'[(1,false,true), (2,true,false), (3,false,false)]
Org_Edit_Abs'Session_types 6: 1'[(1,false,true), (2,true,false), (3,false,false)]
Org_Edit_Abs'Special_Cases 1: 1'SC((1,4))++1'SC((2,0))++1'NSC(())
Org_Edit_Abs'Special_Cases 2: 1'SC((1,4))++1'SC((2,0))++1'NSC(())
Org_Edit_Abs'Special_Cases 3: 1'SC((1,4))++1'SC((2,0))++1'NSC(())
Org_Edit_Abs'Special_Cases 4: 1'SC((1,4))++1'SC((2,0))++1'NSC(())
Org_Edit_Abs'Special_Cases 5: 1'SC((1,4))++1'SC((2,0))++1'NSC(())
Org_Edit_Abs'Special_Cases 6: 1'SC((1,4))++1'SC((2,0))++1'NSC(())
Org_Edit_Abs'Topics 1: 1'1++1'2++1'3++1'4

Org_Edit_Abs'Topics 2: 1'1++1'2++1'3++1'4
Org_Edit_Abs'Topics 3: 1'1++1'2++1'3++1'4
Org_Edit_Abs'Topics 4: 1'1++1'2++1'3++1'4
Org_Edit_Abs'Topics 5: 1'1++1'2++1'3++1'4
Org_Edit_Abs'Topics 6: 1'1++1'2++1'3++1'4
RequestID'Abs_Count 1: empty
RequestID'Abstract_Ids 1: empty
RequestID'Contact_Person 1: 1'{id=6, first_name="", last_name="", email=""}
RequestID'Session_types 1: 1'[(1,false,true), (2,true,false), (3,false,false)]
RequestID'Topics 1: 1'1++1'2++1'3++1'4
Review'Abs_Rev 1: empty
Review'Settings 1: 1'{reviewer_select=true,accepted=3}
Select_New_Reviewer'Abs_Rev 1: empty
Select_New_Reviewer'Reviewers 1: empty
Submit'Abs_Rev 1: empty
Submit'Abs_Rev 2: empty
Submit'Settings 1: 1'{reviewer_select=true,accepted=3}
Submit'Settings 2: 1'{reviewer_select=true,accepted=3}
Submit'Topics_Reviewers 1: empty
Submit'Topics_Reviewers 2: empty

B.3 Mallista analysoidut kotitilaominaisuudet

Home Properties	
Home Markings	None

B.4 Mallista analysoidut elävyysominaisuudet

Liveness Properties	
Dead Markings	None
Dead Transition Instances	None
Live Transition Instances	None

B.5 Mallista analysoidut reiluusominaisuudet

Fairness Properties	
Page'Transition	Fairness
Contact_Edit_Abs'Contact_Edit 1	No Fairness
Contact_Modify_Abs'Contact_Modify 1	No Fairness
Decide'New_Decision0 1	No Fairness
Decide'New_Decision1 1	No Fairness
Decide'New_Decision2 1	No Fairness
Decide'New_Decision3 1	No Fairness
Decide'New_Decision4 1	No Fairness
Decide'New_Decision5 1	No Fairness
Invite_Abs'Invite_Abs 1	Fair
Invite_By_Org'Invite_By_Org 1	Fair
Invite_By_Ses_Org'Invite_By_Ses_Org 1	Fair
Make_Decision'Make_Decision 1	Fair
Org_Edit_Abs'Org_Edit_Abstract 1	No Fairness
Org_Edit_Abs'Org_Edit_Abstract 2	No Fairness
Org_Edit_Abs'Org_Edit_Abstract 3	Just
Org_Edit_Abs'Org_Edit_Abstract 4	No Fairness
Org_Edit_Abs'Org_Edit_Abstract 5	No Fairness
Org_Edit_Abs'Org_Edit_Abstract 6	No Fairness
RequestID'Request_ID 1	Fair
Review'Review 1	No Fairness
Select_New_Reviewer'Select_New_Reviewer 1	Fair
Submit'Submit 1	No Fairness
Submit'Submit 2	No Fairness