T-79.179 Rinnakkaiset ja hajautetut digitaaliset järjestelmät

Aikalogiikka

Marko Mäkelä

23. helmikuuta 2004

Aikalogiikka

Yksinkertaisiltakin näyttävien järjestelmien saavutettavuusgraafeissa on helposti tuhansia tai jopa miljoonia solmuja ja kaaria. Niitä ei ole mielekästä esittää graafisesti saati tutkia käsin solmu solmulta tai kaari kaarelta.

Turvallisuusominaisuudet, kuten järjestelmän lukkiutumattomuus tai kielletyn tilan saavuttamattomuus, on helppo esittää yksittäisiä tiloja koskevina ehtoina, jotka tietokone voi tarkistaa lisätessään tiloja saavutettavuusgraafiin.

Elävyysominaisuuksien ("järjestelmässä tapahtuu edistystä") esittämiseen ja käsittelemiseen tarvitaan uusia menetelmiä, kuten *aikalogiikkaa* ja *mallintarkastusmenetelmiä*.

Järjestelmien ominaisuuksia

- turvallisuus *(safety)* : "järjestelmä ei joudu pahaan tilaan"; kussakin tilassa pätee p
  - lukkiutumattomuus *(deadlock freedom)*
  - keskinäinen poissulkevuus *(mutual exclusion)* jne.
- elävyys *(liveness)* : "järjestelmä etenee"; X tapahtuu äärettömän usein
- reiluus *(fairness)* ; X:n jälkeen Y tapahtuu n askeleen kuluessa
  - lähetetyt viestit saapuvat perille
  - kukin palvelupyyntö suoritetaan
- vakautuvuus *(self-stabilisation)* : "järjestelmä toipuu häiriöstä äärellisessä ajassa"

Ajan kuvaaminen

Useimmat edellä esitetyt aikaan liittyvät ominaisuudet voidaan kuvata yhdistelemällä kahta operaatiota:

- *joskus* tulevaisuudessa
- *aina* tulevaisuudessa

On valittava, kuuluuko nykyhetki tulevaisuuteen. Aikaa voidaan kuvata monella tavalla:

- yhteisenä tai kullekin osapuolelle erillisenä
- lineaarisena tai haarautuvana
- portaittaisena tai jatkuvana

Voidaan kuvata joko siirtymien ajankohtia tai järjestelmän tilaa kunakin hetkenä.

Ajan kuvaaminen: valinnat

- Kaikille osapuolille yhteinen aika etenee portaittain siirtymien mukaan.
- Nykyhetki kuuluu tulevaisuuteen.
- Pääsääntöisesti tarkastellaan tiloja ajan funktiona, ei tapahtumia.

Yksi asia on vaikea ratkaista: käytetäänkö lineaarista vai haarautuvaa aikaa?

LTL *(linear temporal logic)* : suoritukset esitetään *äärettöminä* siirtymäjonoina

CTL *(computational tree logic)* : suoritukset esitetään *äärettömänä* siirtymäpuuna

Kummallakin logiikalla voidaan esittää ominaisuuksia, jotka eivät ole esitettävissä toisella logiikalla. **LTL**:n ja **CTL**:n yhdistelmä **CTL*** on vielä *ilmaisuvoimaisempi*: sillä voidaan esittää ominaisuuksia, joihin **CTL**:n ja **LTL**:n ilmaisuvoima ei riitä.

Lineaarisen ajan logiikka LTL

- Tilapropositiot eli -kaavat Φ: $p \in \Phi$, jos p kuvaa järjestelmän tilat totuusarvoiksi.
- Kaavat $Fma(\Phi) \supset \Phi$ sisältävät tilakaavojen lisäksi
  - epätoden $\bot \in Fma(\Phi)$
  - implikaation: jos $a \in Fma(\Phi)$ ja $b \in Fma(\Phi)$, niin $a \to b \in Fma(\Phi)$, sekä
  - konnektiivin "aina tulevaisuudessa": jos $a \in Fma(\Phi)$, niin $\Box a \in Fma(\Phi)$.

Muut konnektiivit voidaan määritellä näiden avulla:

$$\begin{array}{ll} \neg a \Leftrightarrow a \to \bot & \Diamond a \Leftrightarrow \neg\Box\neg a \\ a \vee b \Leftrightarrow (\neg a) \to b & \top \Leftrightarrow \neg\bot \\ a \wedge b \Leftrightarrow \neg(a \to \neg b) & \end{array}$$

Tämä on vain yksi tapa määritellä LTL ja sen peruskonnektiivit.

Lineaarisen ajan logiikka LTL: esimerkkejä

- Viesti saapuu aina perille: $\Box((l = 0) \to \Diamond(l = 1))$
- Filosofit 1 ja 2 eivät voi syödä samanaikaisesti: $\Box\neg(\text{syö}_1 \wedge \text{syö}_2)$
- p pätee vain äärellisen monessa tilassa eli $\neg p$ pätee äärellisen siirtymäjakson päätyttyä aina: $\Diamond\Box\neg p$
- p pätee äärettömän monessa tilassa: $\Box\Diamond p$
- Turvallisuusominaisuus: $\Box p$

Lineaarisen ajan logiikan LTL tulkitseminen (1/3)

- LTL-kaavojen tulkinta on määritelty äärettömille tilajonoille.
- Tilajoukon S ja saavutettavuusrelaation $R \subseteq S \times S$ pari $\langle S, R\rangle$ muodostaa *kehyksen*.
- Kullakin tilalla on vain yksi seuraaja: $R : S \to S$
- Relaation $R \subseteq S \times S$ refleksiivinen ja transitiivinen sulkeuma $R^* \supseteq R$ määritellään niin, että se on pienin seuraavat ehdot toteuttava R:n sisältävä joukko:
  - $\bigcup_{s \in S} \langle s, s\rangle \subseteq R^*$ (refleksiivisyys)
  - $\{\langle s,t\rangle, \langle t,u\rangle\} \subseteq R^* \Rightarrow \langle s,u\rangle \in R^*$ (transitiivisuus)
- Φ-*malli* on kolmikko $M = \langle S, R, T\rangle$, missä $T : \Phi \to 2^S$ kuvaa tilakaavat tilajoukoiksi: $T(p)$ on niiden tilojen joukko, joissa p pätee.

Lineaarisen ajan logiikan LTL tulkitseminen (2/3)

- Kaavan a pätevyys mallin M tilassa $s \in S$ ($M \models_s a$) määritellään induktiivisesti:

$$\begin{aligned} M &\models_s p \Leftrightarrow s \in T(p) \\ M &\not\models_s \bot \\ M &\models_s (a \to b) \Leftrightarrow (M \models_s a) \to (M \models_s b) \\ M &\models_s \Box a \Leftrightarrow \forall t \in S, \langle s,t \rangle \in R^* : M \models_t a \end{aligned}$$

- Sanotaan, että kaava a pätee mallissa M (kirjoitetaan $M \models a$), jos ja vain jos se pätee mallin tilajonon ensimmäisessä tilassa $s_0 \in S$: $M \models_{s_0} a$.

Lineaarisen ajan logiikan LTL tulkitseminen (3/3)

- Saavutettavuusgraafin $G = \langle V, E, v_0 \rangle$ juurisolmusta $v_0 \in V$ lähtevät jonot $v_0, v_1, \ldots$, joille $\langle v_i, v_{i+1} \rangle \in E$, ovat malleja.
- Merkitään kaikkien näiden tilajonojen joukkoa $\mathcal{M}(G)$.
- Kaava $a \in Fma(\Phi)$ pätee saavutettavuusgraafissa G, jos ja vain jos $M \models_{v_0} a$ kaikille $M \in \mathcal{M}(G)$.
- Koska silmukkoja sisältävää saavutettavuusgraafia voi vastata ääretön määrä malleja, tämä ei ole mielekäs tapa tarkastaa ominaisuuksien pätevyyttä.
- Seuraavassa kuvataan pääpiirteittäin erästä menetelmää [2] LTL-kaavan pätevyyden tarkistamiseksi.

Esimerkkejä LTL-kaavoista ja vastaavista tilajonoista

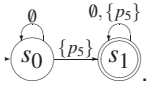
Seuraavassa taulukossa on esitetty kolme ääretöntä tilajonoa, joissa on kuvattu kussakin tilassa pätevät propositiot. Tilan 6 jälkeen propositioiden arvot eivät muutu.

Päteviä kaavoja	0	1	2	3	4	5	6	…
$\Diamond p, \Diamond\neg p, \Diamond\Box p, \Box\Diamond p$		p		p		p	p	…
$\neg\Box\Diamond p, \Box\Diamond q, \Diamond(p \to \Box q)$			p,q	q	q	p,q	q	…
$\neg\Diamond\Box p, \Box\Diamond q, \Box(p \to \Diamond q)$		p	q		p		q	…

Lyhyitä kaavoja voi tarkastaa pienissä malleissa ilman tietokoneen apua tulkitsemalla yhtä aikakonnektiivia kerrallaan aloittamalla aina uuden "silmukan", kun vastaan tulee $\Box$ tai $\Diamond$.

Esimerkiksi kaava $\Diamond(p \to \Box q)$ sanoo, että jos p joskus pätee, myös q pätee kyseisessä tilassa ja kaikissa sitä seuraavissa tiloissa. Propositio p on tosi ainoastaan tiloissa 2 ja 5, ja molemmissa pätee $\Box q$.

Mallintarkastus (1/4)

- Jos tilajoukko on äärellinen, malli voidaan esittää kuvauksena $\mathbb{N} \to (\Phi \to \{\top, \bot\})$ eli $\mathbb{N} \to 2^\Phi$, joka vastaa ääretöntä jonoa merkkejä $m \in 2^\Phi$.
- LTL-kaava a määrittelee erään äärettömien jonojen kielen: niiden mallien joukon, joissa a pätee.
- Esimerkiksi $\Diamond p_5$ pätee kaikissa niissä jonoissa, joissa p_5 pätee joskus. Kielen sanojen aakkosto on $\Sigma = \{\emptyset, \{p_5\}\}$ ja sanat sisältävät ainakin yhden p_5:n.
- Kielen $\Diamond p_5$ sanoja voidaan tunnistaa tilakoneen avulla: 

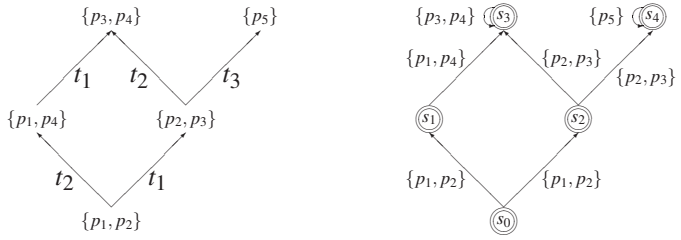
. *Alkutila* on s_0, ja s_1 on *hyväksyvä tila*.

Mallintarkastus (2/4)

- Koska sanat ovat äärettömiä, ei voi soveltaa äärellisten tilakoneiden hyväksymisehtoa (sana hyväksytään, jos tilakone on hyväksyvässä tilassa sanan loppuessa).
- Tilakone Q hyväksyy äärettömän sanan $w = a_0, a_1, \ldots$, jos ja vain jos
  - on ääretön jono $\sigma = s_0, s_1, \ldots$ siten, että s_0 on Q:n alkutila,
  - jokainen s_{i+1} saadaan tilasta s_i ja aakkosesta a_i siirtymäfunktion avulla, ja
  - on ääretön määrä kokonaislukuja $i \in \mathbb{N}$ siten, että s_i on hyväksyvä tila.
- Tilakonetta Q kutsutaan *Büchi-tilakoneeksi*.
- Jokainen LTL-kaava a voidaan kääntää Büchi-tilakoneeksi $B(a)$, joka hyväksyy täsmälleen ne äärettömät tilajonot, joille a pätee.

Mallintarkastus (3/4)

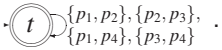
- Saavutettavuusgraafi G voidaan tulkita Büchi-tilakoneena $B(G)$, jossa on pelkkiä hyväksymistiloja ja jossa siirtymät on nimetty lähtötilassaan pätevillä tilakaavoilla.



- Vastaus siihen, päteekö kaava a saavutettavuusgraafissa G saadaan laskemalla tilakoneiden $B(\neg a)$ ja $B(G)$ leikkaus eli tulo. Kaava pätee, jos leikkaustilakone on tyhjä.

Mallintarkastus (4/4)

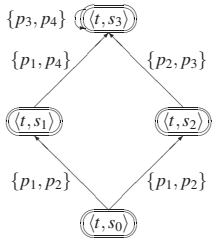
Tarkastetaan kaava $a = \Diamond p_5$ edellisen kalvon saavutettavuusgraafissa. Muodostetaan $B(\neg a) = B(\neg\Diamond p_5) = B(\Box\neg p_5)$:

eli edellisen kalvon $B(G)$:n aakkosia käyttäen


.

Lasketaan tulo $B(\neg a) \times B(G)$. Kas mokomaa, se ei olekaan tyhjä!
Kaavalle $\Diamond p_5$ on kaksi *vastaesimerkkiä*:

$$\{p_1, p_2\} \xrightarrow{t_1} \{p_2, p_3\} \xrightarrow{t_2} \{p_3, p_4\} \to \{p_3, p_4\} \to \cdots$$
$$\{p_1, p_2\} \xrightarrow{t_2} \{p_1, p_4\} \xrightarrow{t_1} \{p_3, p_4\} \to \{p_3, p_4\} \to \cdots$$



Alkuperäisessä mallissa tila $\{p_3, p_4\}$ on lukkiuma, mutta Büchi-automaattiin on tehty silmukka, jotta suoritukset olisivat äärettömiä.

MARIA-työkalu mallintarkastimena

MARIA-työkalu tarkastaa ominaisuuksien pätevyyden samalla, kun se muodostaa mallin saavutettavuusgraafia. Jos annettu ominaisuus ei päde, MARIA esittää sille *vastaesimerkin* eli sellaisen alkutilasta lähtevän siirtymäketjun, jolle ominaisuus ei päde. Aikaa säästyy siihen verrattuna, että saavutettavuusgraafia ruvettaisiin tutkimaan vasta sitten, kun se on saatu kokonaan muodostetuksi.

Yksinkertaisimpien turvallisuusominaisuuksien tarkastamiseen ei kannata käyttää LTL:n raskasta koneistoa. MARIA-kielen `reject`-kaavalla voidaan esittää kielletyt tilat ehtolauseena. Lukkiumien tunnistamista varten voi kirjoittaa `deadlock`-kaavan.*

Tarkastettavat LTL-kaavat syötetään MARIA-kyselykielellä eikä osana mallia, niin kuin `reject`- ja `deadlock`-kaavat.

*Koska LTL käsittelee *äärettömiä* suorituksia, se ei tunne lukkiuman käsitettä. Lukkiumatilat lakaistaan maton alle kuvittelemalla niihin samaan tilaan johtavat silmukat.

LTL:n laajennuksia (1/2)

Joskus on tarvetta puhua aikaväleistä: "ennen kuin", "kunnes":

$$M \models_s (a \mathsf{U} b) \Leftrightarrow \exists \langle s,t \rangle \in R^* : M \models_t b \wedge \\ \forall u \in S, \langle s,u \rangle \in R^*, \langle u,t \rangle \in R^* : (M \models_u a \vee u = t)$$

Vaatimus siitä, ettei vastausta v tule ennen kysymystä k, voidaan kirjoittaa $\Diamond v \rightarrow (\neg v) \mathsf{U} k$.

Kaava $(\neg q \mathsf{U} p) \vee \Box p$ tarkoittaa, että sellaista tilaa, jossa q pätee, pitää edeltää sellainen tila, jossa p pätee.

Konnektiivi U on konnektiiveja $\Diamond$ ja $\Box$ "vahvempi", koska

$$\Diamond a \Leftrightarrow \top \mathsf{U} a \quad \text{ja}$$
$$\Box a \Leftrightarrow \neg(\top \mathsf{U} \neg a).$$

LTL:n laajennuksia (2/2)

Usein tarvitaan konnektiivia "seuraavassa tilassa":

$$M \models_s \bigcirc a \Leftrightarrow M \models_t a, \text{ missä } \langle s,t \rangle \in R$$

Tälle konnektiiville pätee:

$$\Box a \Leftrightarrow a \wedge \bigcirc \Box a$$
$$\Diamond a \Leftrightarrow a \vee \bigcirc \Diamond a$$
$$a \mathsf{U} b \Leftrightarrow b \vee \bigcirc (a \mathsf{U} b)$$

Näiden avulla saadaan seuraavat "laskentasäännöt":

$$\Diamond \Box \neg a \Leftrightarrow \Box \neg a \vee \bigcirc \Diamond \Box \neg a \\ \Leftrightarrow (\neg a \wedge \bigcirc \Box \neg a) \vee \bigcirc \Diamond \Box \neg a$$

$$\Box \Diamond a \Leftrightarrow \Diamond a \wedge \bigcirc \Box \Diamond a \\ \Leftrightarrow (a \vee \bigcirc \Diamond a) \wedge \bigcirc \Box \Diamond a$$

LTL:n ilmaisuvoiman rajallisuus (1/2)

Olkoon ominaisuus p, joka pätee alkutilasta lähtevien polkujen jokaisessa parillisessa tilassa. Parittomissa tiloissa sen totuusarvosta ei sanota mitään:

		0	1	2	3	4	5	6	7	8	9	10	…
s_1	=	p	$\neg p$	p	$\neg p$	p	$\neg p$	p	$\neg p$	p	$\neg p$	p	…
s_2	=	p	p	p	$\neg p$	p	$\neg p$	p	$\neg p$	p	$\neg p$	p	…

Esimerkiksi seuraavat polut rikkovat kyseistä vaatimusta vastaan:

		0	1	2	3	4	5	6	7	8	9	10	…
s_3	=	$\neg p$	p	$\neg p$	p	$\neg p$	p	$\neg p$	p	$\neg p$	p	$\neg p$	…
s_4	=	p	$\neg p$	$\neg p$	$\neg p$	p	$\neg p$	$\neg p$	$\neg p$	p	$\neg p$	p	…

LTL:n ilmaisuvoiman rajallisuus (2/2)

Esimerkiksi seuraavat kaavat eivät erottele esitettyjä polkuja halutulla tavalla:

$$p \wedge \Box(p \rightarrow \bigcirc \neg p) \wedge \Box(\neg p \rightarrow \bigcirc p)$$
$$p \wedge \Box(p \rightarrow \bigcirc \bigcirc p)$$

Toinenkin, hieman yleisempi tulos pätee:

Sellaisella kaavalla, jossa on tilakaavoja ja enintään n konnektiivia $\bigcirc$, on sama totuusarvo kaikille jonoille

$$\sigma_k = \langle \underbrace{p, p, \ldots p}_{k \text{ kpl}}, \neg p, p, p, \ldots \rangle,$$

missä $k > n$.

Viitteitä

1. Z. Manna ja A. Pnueli: *The Temporal Logic of Reactive and Concurrent Systems: Specification*. Aikalogiikan käyttäminen järjestelmien spesifiointiin; tarkistaminen todistamalla.

2. P. Wolper: *Temporal Logic*. Chapter 4 in A. Thayse, *From Modal Logic to Deductive Databases*. Mallintarkastusalgoritmit.

3. J. Esparza ja S. Merz: *Model Checking* (luentokalvot). Johdanto LTL:ään, CTL:ään ja mallintarkastusalgoritmeihin.

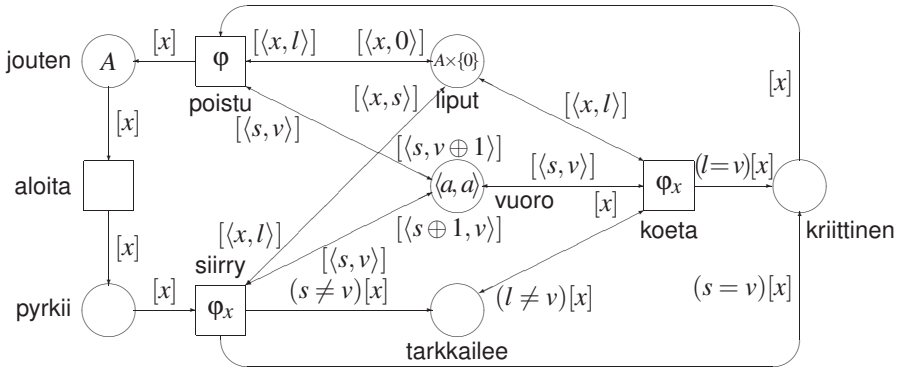
Esimerkki: keskinäisen poissulkevuuden algoritmi

Tarkastetaan aikalogiikalla seuraava algoritmi:

Pääsylippualgoritmi: Jaetussa muuttujassa on pari $\langle seuraava, vuorossa \rangle$, jonka osat voivat saada arvoja $\{1, \ldots, n\}$, alussa $\langle 1, 1 \rangle$. Parin ensimmäinen puolisko esittää seuraavaa "pääsylippua" kriittiseen lohkoon, ja toinen sitä lippua, jolla on viimeksi päässyt kriittiseen lohkoon. Kun prosessi aloittaa kriittiseen lohkoon pyrkimisen, se ottaa "jonotusnumeron" tai "pääsylipun" lukemalla puoliskon *seuraava* ja kasvattamalla sitä **mod** n. Prosessi pääsee kriittiseen lohkoon, kun sen hallitseman lipun numero täsmää *vuorossa*-komponentin kanssa. Poistuessaan kriittisestä lohkosta prosessi kasvattaa muuttujaa *vuorossa* **mod** n.

Nancy A. Lynch: Distributed Algorithms, 1996, ISBN 1-55860-348-4

Pääsylippualgoritmin korkean tason verkko



Tarkastettavia pääsylippualgoritmin ominaisuuksia

Olkoon G pääsylippualgoritmin kahden prosessin mallin ($A = \{1,2\}$) saavutettavuusgraafi. Siitä voidaan tarkastaa seuraavat ominaisuudet:

- Kriittisessä lohkossa on enintään yksi prosessi: $G \models \Box\neg(\text{kriittinen}(1) \wedge \text{kriittinen}(2))$
- On olemassa suoritus, jossa prosessi pääsee kriittiseen lohkoon (haetaan vastaoletuksen vastaesimerkki): $G \not\models \Box\neg\text{kriittinen}(1)$ tai $G \not\models \Box\neg\text{kriittinen}(2)$
- Jos prosessi haluaa kriittiseen lohkoon, se pääsee sinne lopulta (tämä ei päde!): $G \models \Box(\text{pyrkii}(1) \rightarrow \Diamond\text{kriittinen}(1))$ tai $G \models \Box(\text{pyrkii}(2) \rightarrow \Diamond\text{kriittinen}(2))$.

Viimeinen ominaisuus ei päde esimerkiksi sellaisessa suorituksessa, jossa toinen prosessi ensin pääsee kriittiseen lohkoon ja toinen "jää rannalle". Nyt, jos kriittiseen lohkoon päässyt prosessi poistuu ja yrittää uudelleen, se ei pääse etenemään, ellei rannalle jääneen prosessin suorittamista jatketa. Tarvitaan *reiluusoletuksia*.

Reiluus (1/3)

Jos järjestelmä voi edetä loputtomiin suorittamatta jotakin vireessä olevaa siirtymää, järjestelmän saavutettavuusgraafissa on sellaisia äärettömiä polkuja, joissa kyseistä siirtymää "sorretaan"loputtomasti. Näiden asioiden kuvaamiseksi on kehitetty *reiluuden* käsitteitä äärettömän mittaisille suorituksille:

heikko reiluus: jatkuvasti virittynyt siirtymä laukeaa äärettömän usein ($\Diamond\Box v \rightarrow \Box\Diamond l$)

vahva reiluus: äärettömän usein vireeseen tuleva (mahdollisesti silloin tällöin vireestä poistuva) siirtymä laukeaa äärettömän usein ($\Box\Diamond v \rightarrow \Box\Diamond l$)

Yksittäinen reiluusoletus voi koskea yhtä siirtymää tai siirtymäjoukkoa. Jälkimmäisessä tapauksessa edellytetään, että jos siirtymäjoukossa on aina (tai äärettömän usein) virittyneitä siirtymiä, jokin joukkoon kuuluva siirtymä laukaistaan äärettömän usein.

Jos tehdään useita reiluusoletuksia, sallittujen vastaesimerkkien on kohdeltava kunkin reiluusjoukon siirtymiä reilusti edellä mainitulla tavalla.

Reiluus (2/3)

MARIA-työkalussa on mahdollista määritellä reiluusjoukkoja sekä saman siirtymän eri muuttujasidonnoille että useiden siirtymien erilaisille muuttujasidonnoille. Jotta kaavat $G \models \Box(\text{pyrkii}(x) \rightarrow \Diamond\text{kriittinen}(x))$ pätisivät kaikille $1 \leq x \leq n$, tarvitaan $n+1$ heikkoa reiluusoletusta, missä $n = |A|$ on prosessien määrä.

Joukko φ: Siirtymälle "poistu" on oletettava heikko reiluus. Kaikki siirtymän muuttujasidonnat kuuluvat samaan joukkoon. Toisin sanoin: jos siirtymä "poistu" on aina vireessä, se on suoritettava äärettömän usein.

Joukot φ_x: Siirtymille "siirry" ja "koeta" on oletettava heikko reiluus kullekin muuttujan x sidonnalle. Toisin sanoin: jos prosessin x ($1 \leq x \leq n$) on jatkuvasti mahdollista suorittaa joko "siirry" tai "koeta", sen on tehtävä niin äärettömän usein.

Reiluus (3/3)

Reiluusoletukset on luontevinta määritellä suoraan tapahtumajoukoille. MARIA on ensimmäinen korkean tason verkkojen työkalu, jossa niin voi tehdä. Perinteisesti reiluusoletukset on täytynyt esittää osana tarkastettavaa kaavaa, esimerkiksi

$$((\Diamond\Box v_1 \rightarrow \Box\Diamond l_1) \vee \cdots \vee (\Diamond\Box v_n \rightarrow \Box\Diamond l_n)) \rightarrow \Box(\text{pyrkii}(x) \rightarrow \Diamond\text{kriittinen}(x)).$$

Tämä on äärimmäisen tehotonta kahdestakin syystä:

1. LTL-kaavasta syntyvässä Büchi-tilakoneessa on pahimmassa tapauksessa eksponentiaalinen määrä tiloja ja kaaria kaavan konnektiivien määrään nähden.
2. Tarkastettavaa mallia on usein täydennettävä: voidaan tarvita ylimääräisiä paikkoja (ja pahimmassa tapauksessa moninkertaisesti tiloja saavutettavuusgraafiin), jotta
   - kyetään esittämään vaatimus siirtymän laukeamisesta sekä
   - mallintamaan vuorontaja *(scheduler).*

Turvallisuusominaisuuksista

Elävyysominaisuuksista on mielekästä puhua vain, kun suoritukset ovat äärettömän mittaisia eli järjestelmän saavutettavuusgraafissa on silmukoita (tai graafi on ääretön).

Turvallisuusominaisuuksia on huomattavasti kevyempää tutkia ilman Büchi-tilakoneita. Aina, kun löydetään uusi tila, voidaan tarkastaa, täyttääkö se vaatimukset.

Joskus voidaan suoraan hyödyntää mallinnuskielen virheenkäsittelyrutiineja. Esimerkiksi kuvattaessa pääsylippualgoritmi MARIA-kielellä voidaan asettaa paikan "kriittinen" sallituksi merkkimääräksi 0..1, jolloin määrän ylityksestä tulostuu virheilmoitus.

LTL:llä voidaan ilmaista muitakin turvallisuusominaisuuksia kuin $\Box p$. LTL:n turvallisuusosajoukko voidaan kääntää tavallisiksi äärellisiksi tilakoneiksi, jolloin mallintarkastus onnistuu pelkkiä tiloja tutkimalla, ilman silmukoiden ja reiluuden tarkastelua.