Predikaattitapahtumaverkot

- Predikaatit paikkoja, tapahtumat transitioita, tapaukset merkintöjä, kaarilla algebrallisia lausekkeita.
- Merkintä määrittelee paikan sisällöksi merkkijoukon.
- Muuttujien ollessa kiinnitettyinä kaarilauseke ilmaisee kaaren suunnan mukaisesti joko poistuvien tai saapuvien merkkien joukon.

Laskostus ja aukikääriminen

- Laskostus (folding): rakennetaan uusien muuttujien avulla verkko, jossa on alkuperäistä verkkoa vähemmän paikkoja tai transitioita mutta jonka saavutettavuusgraafi on isomorfinen alkuperäisen verkon saavutettavuusgraafin kanssa. Esimerkki: ehtotapahtumajärjestelmän laskostaminen predikaattitapahtumaverkoksi.
- Aukikääriminen (unfolding): laskostukselle käänteinen menettelytapa. Jotta alkuperäinen verkko saataisiin aukikäänitystä versiosta laskostamalla, on laskostusalgoritmille käytännössä erikseen kerrottava, miten aukikääriminen tarkalleen ottaen suoritettiin.

8.3[Rei85] vs. PROD (alkuosa)

```
Jos n = 10, on oheisen verkon saavutettavuusgraafi isomorfinen osion 8.3[Rei85]
tarkasteleman verkon saavutettavuusgraafin kanssa, vaikka PROD ei mitenkään
erityisesti tue [Rei85]-tyylisiä predikaattitapahtumaverkkoja.
kvarpaan@kosh.hut.fi 53: tail -n 50 $PRODPATH/sample/dbm.net
#define n 10
#define rot(x, i) (1 + (((x) + ((i) - 1)) % n))
#place inactive lo(<.1.>) hi(<.n.>) mk(<.1..n.>)
#place waiting lo(<.1.>) hi(<.n.>)
#place performing lo(<.1.>) hi(<.n.>)
#place exclusion mk(<..>)
#place unused lo(<.1, 1.>) hi(<.n, n.>) \
    mk(<.1..n, 1..n.> - (<.1, 1.> + <.2, 2.> + <.3, 3.> + <.4, 4.> + \
                         <.5, 5.> + <.6, 6.> + <.7, 7.> + <.8, 8.> + \
                         <.9, 9.> + <.n, n.>))
#place sent lo(<.1, 1.>) hi(<.n, n.>)
#place received lo(<.1, 1.>) hi(<.n, n.>)
#place acknowledged lo(<.1, 1.>) hi(<.n, n.>)
#trans update_and_send_messages
    in  { inactive: <.s.>;
          exclusion: <..>;
          unused: <.s, rot(s, 1).> + <.s, rot(s, 2).> + <.s, rot(s, 3).>
                + <.s, rot(s, 4).> + <.s, rot(s, 5).> + <.s, rot(s, 6).>
                + <.s, rot(s, 7).> + <.s, rot(s, 8).> + <.s, rot(s, 9).>; }
    out { waiting: <.s.>;
          sent: <.s, rot(s, 1).> + <.s, rot(s, 2).> + <.s, rot(s, 3).>
              + <.s, rot(s, 4).> + <.s, rot(s, 5).> + <.s, rot(s, 6).>
              + <.s, rot(s, 7).> + <.s, rot(s, 8).> + <.s, rot(s, 9).>; }
#endtr
```


8.3[Rei85] vs. PROD (loppuosa)

```
#trans receive_acknowledgements
    in  { waiting: <.s.>;
          acknowledged: <.s, rot(s, 1).> + <.s, rot(s, 2).> + <.s, rot(s, 3).>
                      + <.s, rot(s, 4).> + <.s, rot(s, 5).> + <.s, rot(s, 6).>
                      + <.s, rot(s, 7).> + <.s, rot(s, 8).>
                      + <.s, rot(s, 9).>; }
    out { inactive: <.s.>;
          exclusion: <..>;
          unused: <.s, rot(s, 1).> + <.s, rot(s, 2).> + <.s, rot(s, 3).>
                + <.s, rot(s, 4).> + <.s, rot(s, 5).> + <.s, rot(s, 6).>
                + <.s, rot(s, 7).> + <.s, rot(s, 8).> + <.s, rot(s, 9).>; }
#endtr
#trans receive_message
    in  { inactive: <.r.>;
          sent: <.s, r.>; }
    out { performing: <.r.>;
          received: <.s, r.>; }
    gate s != r;
#endtr
#trans send_acknowledgement
    in  { performing: <.r.>;
          received: <.s, r.>; }
    out { inactive: <.r.>;
          acknowledged: <.s, r.>; }
    gate s != r;
#endtr
```