Topics in Concurrency

Lecture 10

Jonathan Hayman

10 November 2016

Petri nets

- Introduced in 1962 (though claimed to have been invented by 1939)
- Starting point: think of a transition system where a number of processes can be in a given state and then allow coordination
- Conditions: local components of state
- Events: transitions and coordination
- Allows study of concurrency of events, reasoning about causal dependency and how the action of one process might conflict with that of another
- The first of a range of models: event structures, Mazurkiewicz trace languages, asynchronous transition systems, ...
- Many variants with different algorithmic properties and expressivity

∞-multisets

Multisets generalise sets by allow elements to occur some number of times. ∞-multisets generalise further by allowing infinitely many occurrences.

$$\omega^{\infty} = \omega \cup \{\infty\}$$

Extend addition:

$$n + \infty = \infty \qquad \text{for } n \in \omega^{\infty}$$

Extend subtraction

$$\infty - n = \infty \qquad \text{for } n \in \omega$$

Extend order:

$$n \leq \infty \qquad \text{for } n \in \omega^{\infty}$$

An ∞-multiset over a set X is a function

$$f : X \to \omega^{\infty}$$

It is a multiset if $f : X \to \omega$.

Operations on ∞-multisets

- $f \le g$ iff $\forall x \in X. f(x) \le g(x)$
- $f + g$ is the ∞-multiset such that

$$\forall x \in X.\ (f+g)(x) = f(x) + g(x)$$

- For g a multiset such that $g \le f$,

$$\forall x \in X.\ (f-g)(x) = f(x) - g(x)$$

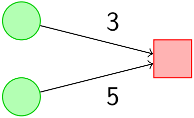
General Petri nets

A general Petri net consists of

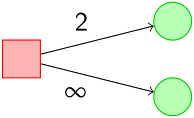
- a set of conditions P

- a set of events T

- a pre-condition map assigning to each event t a multiset of conditions ${}^\bullet t$



- a post-condition map assigning to each event t an ∞-multiset of conditions $t^\bullet$



- a capacity map Cap an ∞-multiset of conditions, assigning a capacity in ω^∞ to each condition

Dynamics

A marking is an ∞-multiset $\mathcal{M}$ such that

$$\mathcal{M} \leq Cap$$

giving how many tokens are in each condition.

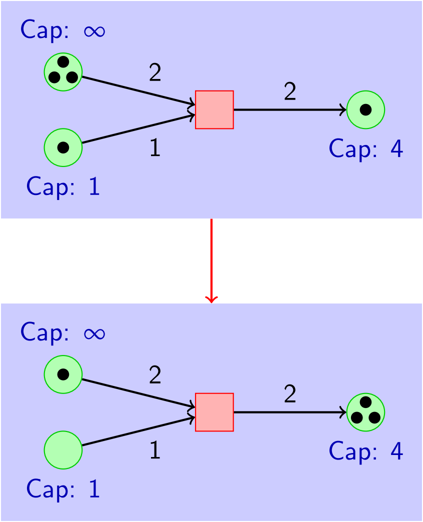
The token game:

For $\mathcal{M}, \mathcal{M}'$ markings, t an event:

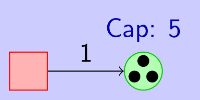
$$\mathcal{M} \xrightarrow{t} \mathcal{M}' \quad \text{iff} \quad {}^{\bullet}t \leq \mathcal{M} \quad \& \quad \mathcal{M}' = \mathcal{M} - {}^{\bullet}t + t^{\bullet}$$

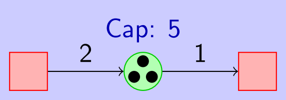
An event t has concession (is enabled) at $\mathcal{M}$ iff

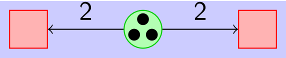
$${}^{\bullet}t \leq \mathcal{M} \quad \& \quad \mathcal{M} - {}^{\bullet}t + t^{\bullet} \leq Cap$$

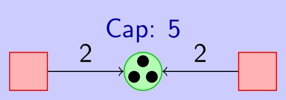

Cap: ∞
2
2
1
Cap: 4
Cap: 1
Cap: ∞
2
2
1
Cap: 4
Cap: 1

Further examples









Basic Petri nets

Often don't need multisets and can just consider sets.

A basic net consists of

- *a set of conditions* B
- *a set of events* E
- *a pre-condition map assigning a subset of conditions* ${}^{\bullet}e$ *to any event* e
- *a post-condition map assigning a subset of conditions* $e^{\bullet}$ *to any event* e *such that*

$$ {}^{\bullet}e \cup e^{\bullet} \neq \emptyset $$

The capacity of any condition is implicitly taken to be 1*:*

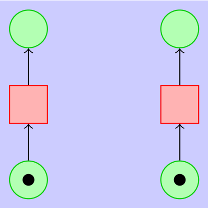
$$ \forall\, b \in B : \; Cap(b) = 1 $$

A marking $\mathcal{M}$ *is now a subset of conditions.*

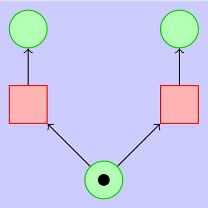
$$ \mathcal{M} \xrightarrow{e} \mathcal{M}' \quad \textit{iff} \quad \begin{array}{rl} & {}^{\bullet}q \subseteq \mathcal{M} \quad \& \quad (\mathcal{M} \setminus {}^{\bullet}e) \cap e^{\bullet} = \emptyset \\ \& & \mathcal{M}' = (\mathcal{M} \setminus {}^{\bullet}e) \cup e^{\bullet} \end{array} $$

Concepts

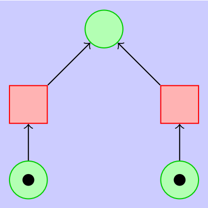
Concurrency



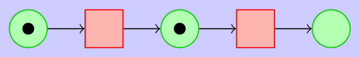
Forwards conflict



Backwards conflict



Contact



Safe nets

- Contact occurs in marking M if there exists an event e such that

$$ {}^{\bullet}e \subseteq M \qquad (M \setminus {}^{\bullet}e) \cap e^{\bullet} \neq \varnothing $$

- A basic net is safe if there is no marking reachable from the initial marking in which contact occurs.

CCS operations on basic nets

A safe Petri net semantics for CCS can be constructed by ‘surgery’ on the nets:

- Nil process
- Prefixing
- $p + q$
- $p \parallel q$