Calculi of Concurrency

Temporal logics for CCS

Ulf Nilsson

2000-05-09

Verification

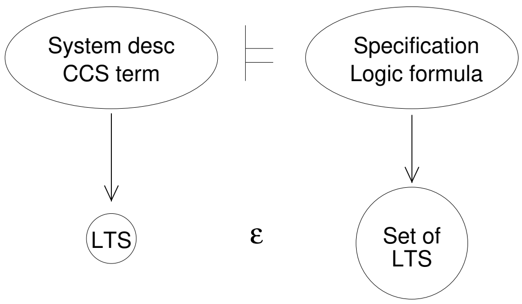
A system is an LTS which can be described by a term in CCS.

Simulations and bisimulations can be used to reason about the system. Specification is expressed in the same formalism.

But we can also use a dedicated notation to express properties of the system.

- mu-calculus (Hennessy-Milner Logic + fixpoints)
- computation tree logic (CTL)

Verification problem for CCS



Properties under consideration

Safety properties: all reachable states are safe. Nothing bad happens.

Liveness properties: some states are always reachable. Something good eventually happens.

Properties involving *temporal order* of *actions*; not explicit time.

Action-based; not state-based; but states can (sometimes) be observed through actions.

Hennessy-Milner Logic

Let $\mathcal{A}$ be a set of action, $a \in \mathcal{A}$ and $K \subseteq \mathcal{A}$.

Abstract syntax:

$$F \ ::= \ tt \mid \neg F \mid F_1 \wedge F_2 \mid [K]F$$

Formulas are interpreted over an LTS:

$$< \mathcal{P}, \{\stackrel{a}{\to}\}_{a \in \mathcal{A}} >$$

Let $E \in \mathcal{P}$, then:

$$\begin{array}{lll} E \models tt & & \\ E \models \neg F & \text{iff} & E \not\models F \\ E \models F_1 \wedge F_2 & \text{iff} & E \models F_1 \text{ and } E \models F_2 \\ E \models [K]F & \text{iff} & \forall E' \in \mathcal{P}\ \forall a \in K \\ & & \text{whenever } E \stackrel{a}{\to} E' \text{ then } E' \models F \end{array}$$

Hennessy-Milner Logic (cont'd)

Some syntactic sugar:

$$\begin{array}{rcl} ff & \equiv & \neg tt \\ F_1 \vee F_2 & \equiv & \neg(\neg F_1 \wedge \neg F_2) \\ \langle K \rangle F & \equiv & \neg [K] \neg F \\ [a_1, \ldots, a_n]F & \equiv & [\{a_1, \ldots, a_n\}] \\ [-K]F & \equiv & [\mathcal{A} - K]F \\ [-]F & \equiv & [\mathcal{A}]F \end{array}$$

Semantics of $\langle K \rangle F$:

$$\begin{array}{lll} E \models \langle K \rangle F & \text{iff} & \exists E' \in \mathcal{P}\ \exists a \in K \\ & & E \stackrel{a}{\to} E' \text{ and } E' \models F \end{array}$$

Elementary properties

Capacity to perform actions:

$$\langle K \rangle tt$$

Inability to perform actions:

$$E \models [K] ff$$

Deadlock:

$$E \models [-] ff$$

Necessity to perform an action:

$$E \models \langle - \rangle tt \wedge [-a] ff$$

Example

The process

$$V \equiv \text{2p.big.collect.}V + \text{1p.little.collect.}V$$

corresponds to an LTS.

Proof of simple property:

1. big.collect.$V \models$ [little]ff
2. big.collect.$V \models \langle$big$\rangle tt$
3. big.collect.$V \models$ [little]$ff \wedge \langle$big$\rangle tt$
4. $V \models$ [2p]([little]$ff \wedge \langle$big$\rangle tt$

Example

Counter:

$$\begin{aligned}\mathsf{CT}_0 &= \mathsf{up}.\mathsf{CT}_1 + \mathsf{around}.\mathsf{CT}_0 \\ \mathsf{CT}_{n+1} &= \mathsf{up}.\mathsf{CT}_{n+2} + \mathsf{down}.\mathsf{CT}_n\end{aligned}$$

Property requiring inductive reasoning:

$$\mathsf{CT}_m \models [\mathsf{up}]^n \langle \mathsf{down} \rangle^n tt$$

Semantics (revisited)

A formula F denotes a set $\|F\|$ of processes; namely $\|F\| = \{E \in \mathcal{P} \mid E \models F\}$:

$$\begin{aligned}\|tt\| &= \mathcal{P} \\ \|\neg F\| &= \mathcal{P} - \|F\| \\ \|F_1 \wedge F_2\| &= \|F_1\| \cap \|F_2\| \\ \|[K]F\| &= \overline{[K]}\, \|F\|\end{aligned}$$

where

$$\overline{[K]}\mathcal{P}' = \{E \in \mathcal{P} \mid \forall a \in K. \text{ if } E \xrightarrow{a} E' \text{ then } E' \in \mathcal{P}'\}$$

Moreover

$$\|\langle K \rangle F\| = \overline{\langle K \rangle}\, \|F\|$$

where

$$\overline{\langle K \rangle}\mathcal{P}' = \{E \in \mathcal{P} \mid \exists E' \in \mathcal{P}' \; \exists a \in K \; E \xrightarrow{a} E'\}$$

"Weak" Hennessy-Milner Logic

Abstract syntax for weak HML:

$$F \;::=\; tt \mid \neg F \mid F_1 \wedge F_2 \mid [\![K]\!]F$$

Formulas are interpreted over an LTS:

$$< \mathcal{P}, \{\stackrel{a}{\Rightarrow}\}_{a \in \mathcal{A}} >$$

Let $E \in \mathcal{P}$, then:

$$\begin{array}{lll} E \models tt & & \\ E \models \neg F & \text{iff} & E \not\models F \\ E \models F_1 \wedge F_2 & \text{iff} & E \models F_1 \text{ and } E \models F_2 \\ E \models [\![K]\!]F & \text{iff} & \forall E' \in \mathcal{P} \; \forall a \in K \\ & & \text{whenever } E \stackrel{a}{\Rightarrow} E' \text{ then } E' \models F \end{array}$$

"Weak" Hennessy-Milner Logic (cont'd)

Derived operator:

$$E \models \langle\!\langle K \rangle\!\rangle F \quad \text{iff} \quad \exists E' \in \mathcal{P} \; \exists a \in K \; E \stackrel{a}{\Rightarrow} E'$$

For instance:

$$a.0 + \tau.b.0 \models \langle\!\langle \epsilon \rangle\!\rangle [\![a]\!].ff$$

but

$$a.0 + b.0 \not\models \langle\!\langle \epsilon \rangle\!\rangle [\![a]\!].ff$$

where $\langle\!\langle \epsilon \rangle\!\rangle$ is the capablity to perform 0 or more reactions.

Recursive equations

HML facilitates reasoning about finite behaviors (or a finite prefix of an infinite behavior). But we can not reason about infinite repetitive behaviors such as

- action a may occur infinitely often
- action b always occurs after a finite sequence of a-actions

Infinite behaviors can be expressed using recursive equations:

$$X = \langle a \rangle X$$

X is a set of processes and the equation says that any $E \in X$ is able to perform the action a and then become a process in X.

Recursive equations (cont'd)

An equation such as $X = \langle a \rangle X$ may have many solutions. E.g. $\emptyset$, or $\{A\}$ where $A = a.A$.

Some other equations have no solutions, e.g. $X = \neg X$. For an equation $X = F(X)$, we know that $F(X)$ is monotonic if every free occurrence of X is within the scope of an even number of negations.

If $F(X)$ is monotonic it is known to have a least and greatest fixpoint; i.e. a $\mathcal{P}'$ such that $\mathcal{P}' = F(\mathcal{P}')$ [Tarski].

We use $\mu X.F(X)$ to denote the least solution to the equation $X = F(X)$.

We use $\nu X.F(X)$ to denote the greatest solution to the equation $X = F(X)$.

The modal mu-calculus

Abstract syntax:

$$F \ ::= \ X \mid \neg F \mid F_1 \wedge F_2 \mid [K]F \mid \nu X.F$$

Intuitively $\nu X.F$ reads “the largest set of tasks X such that $X = F$”.

$$\begin{aligned} \textit{tt} &\equiv \nu X.X \\ \mu X.F &\equiv \neg \nu X.\neg F[X/\neg X] \end{aligned}$$

Some useful properties

An infinite sequence of a-actions:

$$\nu X.\langle a \rangle X$$

Ability to do K-actions until F holds

$$\mu X.F \vee \langle K \rangle X$$

F may eventually hold:

$$\mu X.F \vee \langle - \rangle X$$

Capacity to do infinite sequence of K-actions where F holds throughout.

$$\nu X.F \wedge \langle K \rangle X$$

Computation Tree Logic: CTL

Abstract syntax:

$$\begin{array}{lll} F & ::= & tt \mid ff \mid \neg F \mid F_1 \wedge F_2 \mid F_1 \vee F_2 \mid \\ & & \mathbf{AG}F \mid \mathbf{AF}F \mid \mathbf{EG}F \mid \mathbf{EF}F \mid \\ & & \mathbf{A}(F_1 \ \mathbf{U} \ F_2) \mid \mathbf{A}(F_1 \ \mathbf{W} \ F_2) \mid \\ & & \mathbf{E}(F_1 \ \mathbf{U} \ F_2) \mid \mathbf{E}(F_1 \ \mathbf{W} \ F_2) \end{array}$$

Constructs from CTL and mu-calculus can be mixed.

NOTE: Computations may be finite!

CTL $\subset$ mu-calculus

$\mathbf{AG}F \equiv \nu X.F \wedge [-]X$

$\mathbf{AF}F \equiv \mu X.F \vee ([-]X \wedge \langle - \rangle tt)$

$\mathbf{A}(F_1 \ \mathbf{U} \ F_2) \equiv \mu X.F_2 \vee (F_1 \wedge [-]X \wedge \langle - \rangle tt)$

$\mathbf{A}(F_1 \ \mathbf{W} \ F_2) \equiv \nu X.F_2 \vee (F_1 \wedge [-]X)$

$\mathbf{EG}F \equiv \nu X.F \wedge (\langle - \rangle X \vee [-]ff)$

$\mathbf{EF}F \equiv \mu X.F \vee \langle - \rangle X$

$\mathbf{E}(F_1 \ \mathbf{U} \ F_2) \equiv \mu X.F_2 \vee (F_1 \wedge \langle - \rangle X)$

$\mathbf{E}(F_1 \ \mathbf{W} \ F_2) \equiv \nu X.F_2 \vee (F_1 \wedge (\langle - \rangle X \vee [-]ff))$