CCS : Syntax and Semantics

- Basic (sequential) operators : Nil, Action prefix, Choice (+)

  All the finite trees can be generated

- Names and recursive definitions : $\begin{cases} A_1 = P_1(\bar{A}) \\ \cdots \\ A_k = P_k(\bar{A}) \end{cases}$

  All the finite state processes can be generated

- Parallel composition and abstraction :

  parallel comp. (|) , restriction and relabelling.

  - Synchronization : names (a) and co-names ($\bar{a}$)

    Internal (non-observable) action (τ)

  - Full syntax : General choice ($\sum_{i \in I} P_i$) ;

    Restriction ($P \setminus L$, $L \subseteq A$)

    Relabelling $P[f]$ ($f(\tau) = \tau$, $\overline{f(a)} = f(\bar{a})$)

- CCS - program : $K_i = P_i(\bar{K})$ $\quad\quad$ $P(\bar{K})$

  declarations $\quad\quad$ main program

- Operational semantics : CCS $\longrightarrow$ LTS
- Structural operational semantics : Rules for the operators

  Rule $\dfrac{\text{premises}}{\text{conclusion}}$ conditions

  Finite derivations $\begin{Bmatrix} \text{prove} \\ \text{derive} \end{Bmatrix}$ transitions

  LTS of a process : $\begin{cases} \text{derived processes} \\ \text{continuations} \end{cases}$

  Either finite (state) or infinite

Full LTS of semantics of all the processes that can be defined on a set of declarations

---

- CCS with data

  Basic motivation : $\left\{ \begin{array}{ll} in(x) & \text{Prepared for any input} \\ \hline out(v) & \text{Produces some concrete value} \end{array} \right.$

- Semantics : $in(x) \rightsquigarrow \sum_{v \in I} in(v)$

- Synchronization via $|$ and $\backslash$

  Allowed by $|$ Imposed by $\backslash$

  Once applied it is hidden but it could be indirectly observed by means of additional actions

  The idea is that the external observer accepts any observable actions produced by the process.

  This is equivalent to assume that "the executions" of a processes are not stopped while there is still some executable action : all of them appear in the generated LTS.

  But deadlocks are (of course!) possible

- Semantics (beyond the operational)

  We need to detect "equivalent" behavious

  . Abstraction : states , duplications , internal actions .

  Equivalence relation

  . Compositional : congruence

  All the operators preserve the equivalence

  We can substitute any component by any equivalent one .

  . Trace semantics : loses the choice points

  It "works" only because termination is not observable due to the prefix - property .

  Non - deterministic behavious could be $\begin{cases} \text{mixed} \\ \text{confused} \end{cases}$

  . Strong bisimilarity : only the "compulsory" identifications

  Local similarity repeat forever both ways

  Coinductive definition : the intuitive "inductive" definition would not work on infinite behaviours even (or specially !) for finite state processes .

  Bisimilar : we can show a (possibly infinite) bisimilarity relation proving it .

  Eq. We cannot prove that they are not different !

  . Nice formal properties : $\sim$ is (the greatest !) bisim .

  It is a congruence for CCS .

– Proving bisimilarity and non-bisimilarity.

- Guessing and checking a bisimulation

  If finite can be checked exhaustively

  If infinite must be finitely presented and checked.

- The bisimulation game

  The attacker looks for a proof of non-bisimilarity

  The defender try to balance any move of the attacker

  The "reversity" of the board captures BI-simulation

  - Finite proofs of non-bisimilarity for finitary processes

    They capture a concrete cause of non-equivalence

– Weak bisimilarity

Weak transitions : $\overset{a}{\Longrightarrow} = \begin{cases} a = \tau & \xrightarrow{\tau}{}^{*} \\ a \neq \tau & \xrightarrow{\tau}{}^{*} . \xrightarrow{a} . \xrightarrow{\tau}{}^{*} \end{cases}$

We must consider $\overset{\tau}{\Rightarrow}$ since it captures silent changes of the state

Weak bisimulations are just strong ones on $\Rightarrow$.

We must include $\overset{\tau}{\Rightarrow}{}^{0} = Id$ since when we do nothing we also observe nothing.

All the techniques to work with strong bisimilarity can be applied to weak bisimilarity

It has (similarly) nearly all its good properties

Unfortunately, sometimes $+$ does not preserve $\approx$

We can (easily) slightly strengthen $\approx$ getting $\approx^c$ that is now a congruence relation for CCS.
