The Concurrency Workbench: *making CCS run*

Featuring:
- CWB Edinburgh Version 7.1
- Emacs &
- daVinci 2.1

1. A word about tools
2. Calculus of Communicating Systems (CCS)
3. The modal $\mu$-Calculus
4. Case study: Fairness is a problem
5. Theory for Practice?
A Short Timeline

‘86 conceived from theoretical, educational and practical concerns used both in teaching and industry treatment for CCS, TCCS and SCCS

‘94 \( \approx 20,000 \) lines of SML code, \( \approx 90 \) commands (this is when a systems- and software engineer was hired)

today Version 7.1 available for Solaris and Linux about 800 KB SML source code interface: Emacs & daVinci
Structure of the CWB

things at this level have empty interface: they are used by users, not by developers

things at this level define basic concepts that may be used by higher level modules

things at this level allow higher-level modules to be implemented more easily
What can we do with the CWB?

- define agents
- simulate agents
- check equalities
- check properties from the modal $\mu$-calculus
CCS in the Concurrency Workbench

AGENT A 0 deadlock (nil)
    a.A action prefix
    tau.A silent prefix
    A + A (weak) choice
    A | A parallel composition
    A \ S restriction: actions in S synchronized
    (A) you can use brackets almost as you’d expect

SET S {a, b, c, ...} a collection of actions
Making Coffee

\[
\begin{align*}
\text{not bisimilar} & \quad \text{not weakly bisimilar} \\
\text{cannot match } A & \xrightarrow{\tau} B
\end{align*}
\]
Drinking Coffee

Add a consumer:

Now P and Q are even strongly bisimilar
Drinking Coffee

Add a consumer:

\[
\begin{align*}
&\text{tea} & \text{coffee} & \text{tea} \\
&\text{coffee} & & \text{tea}
\end{align*}
\]

Now \( P \) and \( Q \) are even strongly bisimilar if we require them to synchronize on their actions:

\[ D := \{ \text{tea}, \text{coffee} \} \]
Reminder: Levels of Equivalence

\[ P \sim Q > P + R \sim Q + R \]
\[ P \parallel R \sim Q \parallel R \]

\[ P \approx Q > \alpha.P + R \approx \alpha.Q + R \]
\[ P \parallel R \approx Q \parallel R \]

\[ P \] can produce trace \( \alpha \) \iff \[ Q \] can produce trace \( \alpha \)
Additional (weak) Equalities

\[ a \approx \tau.a \approx a + \tau.a \]

\[ a.(b + \tau.c) \approx a.(b + \tau.c) + a.c \]
Temporal Logics

- μ-calculus
- ATL*
- CTL*
- ATL
- LTL
- CTL
- HM

{ general 'E' and 'A' formulas

- interaction with environment
- lifeness properties
- safety properties

- bisimulation
μ Calculus Syntax in CWB

PROP P T true
F false
¬P negation
P & P conjunction
P | P disjunction
P => P implication
[a,...]P strong necessity
[-a,...]P strong complement necessity
[[a,...]]P weak necessity
⟨a,...⟩P strong possibility
⟨⟨a,...⟩⟩ weak possibility
Special non-labels

\textbf{tau: unobservable action}

- $\text{tau.a.0} \models \tau \langle a \rangle T$
- $\text{tau.a.0} \not\models \langle a \rangle T$
- $\text{tau.tau.a.0} \not\models \tau \langle a \rangle T$
- $\text{tau.a.0} \models \tau \tau \langle a \rangle T$
- $\text{tau.a.0} \models \tau \tau \langle a \rangle T$
Special non-labels

tau: unobservable action

- $\text{tau.a.0} \models <\text{tau}><\text{a}>T$
- $\text{tau.a.0} \not\models <\text{a}>T$
- $\text{tau.tau.a.0} \not\models <\text{tau}><\text{a}>T$
- $\text{tau.a.0} \models <\text{tau}><\text{a}>T$
- $\text{tau.a.0} \not\models <\text{tau}><\text{a}>T$

not allowed

eps: empty observation

- $\text{tau.a.0} \models <\text{eps}><\text{a}>T$
- $\text{a.0} \models <\text{eps}><\text{a}>T$
- $\text{a.b.0} \models <-\text{eps}><\text{b}>T$
Fixpoint Operators

\[ \text{min}(X.P) \quad \text{least fixpoint temporal formula} \]
\[ \text{max}(X.P) \quad \text{greatest fixpoint temporal formula} \]

\[ \text{max}(Z. \varphi \land \neg Z) \quad AG \varphi \quad \text{Invariant} \]
\[ \text{max}(Z. [a]F \land \neg Z) \quad AG [a]F \quad \text{Safety: Never a} \]
\[ \text{min}(Z. <a>T \lor \neg
\neg Z) \quad EF <a>T \quad \text{Eventually a} \]
\[ \text{min}(Z. [-a]F \lor (\neg
\neg T \land \neg Z)) \quad AF (<a>T \land [-a]F) \quad \text{Inevitably a} \]
\[ \text{min}(Z. Q \land (P \land \neg
\neg T \land \neg Z)) \quad P \text{ Until } Q \quad \text{strong until} \]
\[ \text{max}(Z. Q \land (P \land \neg
\neg Z)) \quad P \text{ Wuntil } Q \quad \text{weak until} \]
\[ \text{max}(Z. [a] \text{min}(Y. \neg
\neg T \land [-b]Y) \land \neg Z) \quad AG(a \Rightarrow AF<b>T) \quad \text{Response} \]
What we have not:

- a notion of states or local propositions
- a global store
- an easy way to check, that a $\mu$-formula and intuition coincide
Common Pitfalls

tau and eps

true: \( \langle \varepsilon \rangle \equiv \langle \neg S \rangle \equiv \tau^* \)

false: \( \langle \neg \varepsilon \rangle \equiv \langle S \rangle \)

e.g. \( \tau.a.0 \models \langle \neg \varepsilon \rangle \langle a \rangle T \) but \( \tau.a.0 \not\models \langle S \rangle \langle a \rangle T \)

Modalities in fixed points

\[ \max(\mathcal{Z}.\varphi \land [-]Z) : \varphi \text{ allways} \]

\[ \max(\mathcal{Z}.\varphi \land [S]Z) : \varphi \text{ in all paths excluding } \tau \]

\( S \): set of all observable actions
Living without Propositional Formulas

Problems:

- deadlock properties not preserved
- AF properties fail now
Living without Propositional Formulas

Problems:
- deadlock properties not preserved
- AF properties fail now

Problems:
- $p$ must be unsynchronized
- we destroy one-step properties
- $a, b$ do not stay enabled
Living without Propositional Formulas (2)

Problems:

- introduces deadlocks
- AF properties fail now

Thus: We can augment our model, to make states observable...
but we have to be careful not modify the behaviour!
Living without Memory

CWB does not allow a store as part of a systems state. > We have to model it explicitly

Variable $M$ of type $\text{int}[0..2]$}

Problems:

- Sequential Queries
- Tedious
Weak Fairness

Want: exclude all runs $\Sigma^* a^\omega$

1. Attempt: Always, $b$ will eventually be taken

$$\nu X. \mu Y. (\langle \rangle T \land [-b] Y) \land [-] X$$

2. Attempt: If $a$ is taken $\infty$ often, then so is $b$

$$\mu X. \nu Y. (\langle a \rangle T \lor X) \land [-b] Y$$
**Weak Fairness**

**Want:** exclude all runs $\Sigma^* a^\omega$

1. **Attempt:** Always, $b$ will eventually be taken

$$\nu X. \mu Y. (\langle \neg \rangle T \land \neg b Y) \land \neg X$$

2. **Attempt:** If $a$ is taken $\infty$ often, then so is $b$

$$\mu X. \nu Y. (\langle a \rangle T \lor X) \land \neg b Y$$

**Problem:** We can *express* fairness,
but not add it as an Assumption

‘inevitably, it will beep’ is equivalent to *false.*
AND: our formulas are equivalent to *false.*
**Dekker’s Mutex Algorithm**

**** Agent 1 ****

while true

    b1 := true

while b2 do

    if k = 2 then

        b1 := false

        while k = 2 skip;

        b1 := true

<enter critical region>

<exit critical region>

k := 2

b1 := false
Dekker’s Mutex Algorithm

**** Agent 1 ****
while true
    b1 := true
while b2 do
    if k = 2 then
        b1 := false
        while k = 2 skip;
        b1 := true
    <enter critical region>
    <exit critical region>
k := 2
b1 := false

Want to prove:
Freedom from individual starvation
Problem: Read Loops

(Unfair) loops are possible.

> Freedom from individual starvation requires a fairness assumption.

In order to incorporate a fairness assumption, we introduce additional observable actions \(a, b, c\).
A Detour to (Starvation) Freedom

1. The system is deadlock-free:

   \[ \text{System} \models \nu Z.\leftrightarrow T \land [-] Z \]

2. It is impossible to reach fair loops:

   \[ \forall \text{actions } x : \text{System} \not\models \mu Z. (\nu X. [\![x]\!] F \land [\![x]\!] X) \lor \leftrightarrow Z \]

3. If actions \( x, y \) happen \( \infty \) often, then \( c \) happens \( \infty \) often:

   \[ \text{System} \models \nu Z. \mu X. ([x](\nu Y.([y](\nu W.(X \land [-c] W))) \land [-c] Y) \land [-] Z) \]

   \[ \begin{array}{c}
   \text{strong fairness} \sim^1 \sim^2 \, \text{at least two actions are observed} \infty \text{ often} \\
   \sim^3 \, \text{freedom of individual starvation}
   \end{array} \]
Is the CWB a Tool for Industry?

Motivations for using the CWB

- curiosity (see CCS ’work’)
- verification (prove properties about your model)
- the attractive expressiveness of $\mu$-calculus formulas
- experiments with own – process algebras
  – logics
  – model checking algorithms

Limitations

- SML implementation rather consumptive (time/memory)
- graphical viewer does not scale well

\[\sim\] can be overcome... by investment of sufficient manpower
Why is it not used every day?

- *interface* is a command-line
- the *agent model* is unfamiliar to engineers
- logic is *too difficult* to understand
How do Industrial Tools Look Like?

Industrial tools

- do things that are conceptually simple
- ... but large and complex
- are (relatively) easy to understand and to operate
- have to be capable of dealing with large instances

... and they have nice user interfaces(!)

methods that are difficult to learn

technologies that require experts

hands off: non-proven technologies
Tools in Practice

- calculus
- $\mu$-calculus
- CWB[−NC], Theorem provers
- ATL*
- CTL*
- no specialized tools
- ATL
- Mocha
- LTL
- SMV
-CTL
- HM
- reachability (e.g. visualState)

Incomplete Methods: Simulation, (automated) Testing
Invitation: Dig Deeper

The user manual The Edingburgh Concurrency Workbench (Version 7.1)

Colin Stirling’s Article Bisimulation, Model Checking and other Games

The tool page of Kim’s course http://www.brics.dk/~omoeller/v01/

Find these slides at
http://www.brics.dk/~omoeller/v01/cwb.pdf