

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 μ -Calculus
4. Case study: Fairness is a problem
5. Theory for Practice?

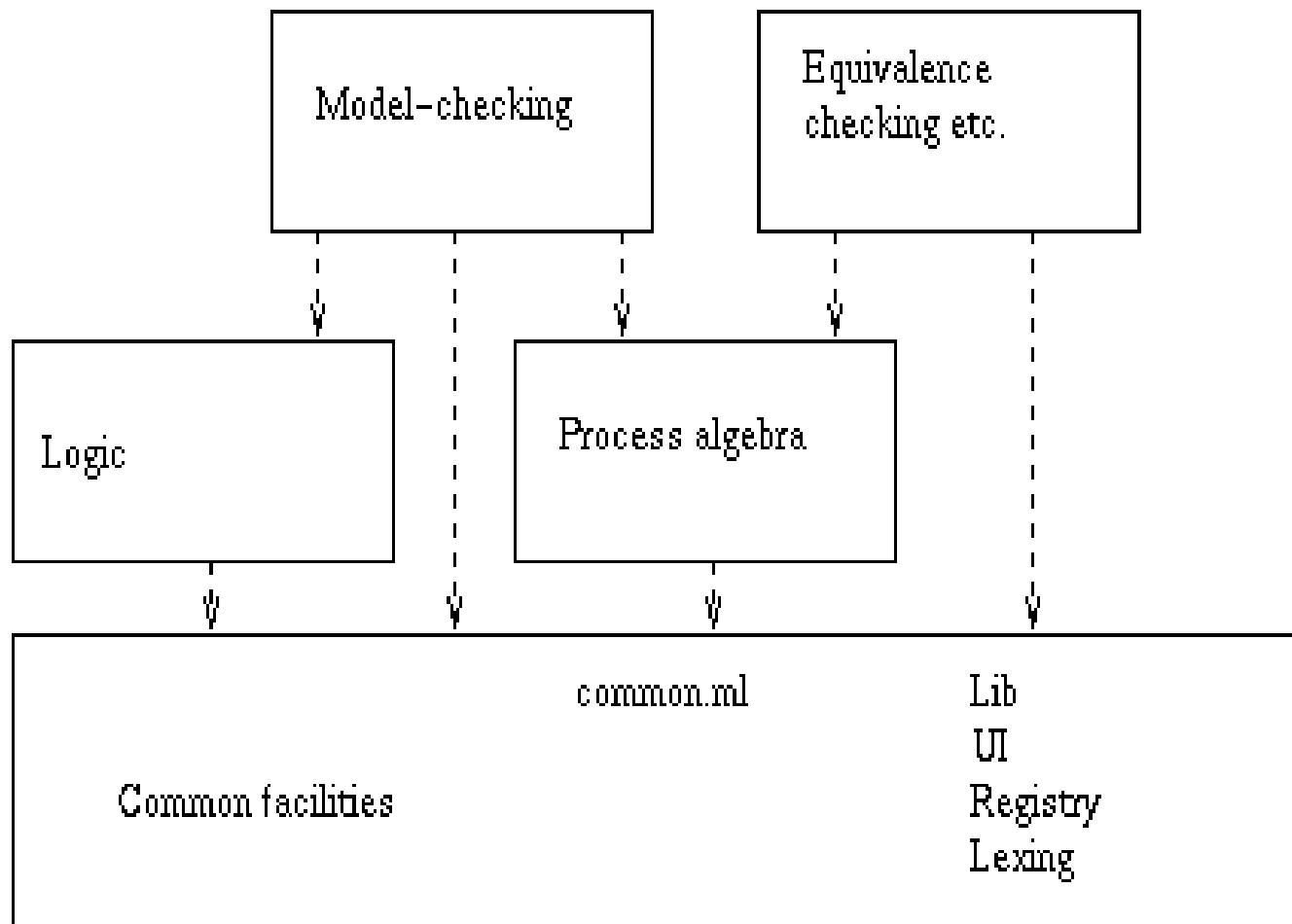
A Short Timeline

'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

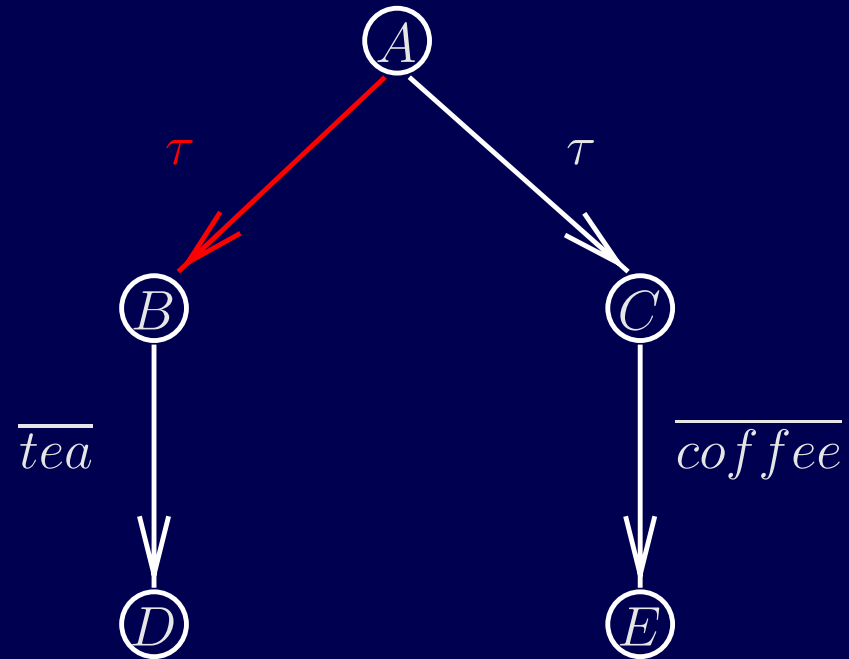
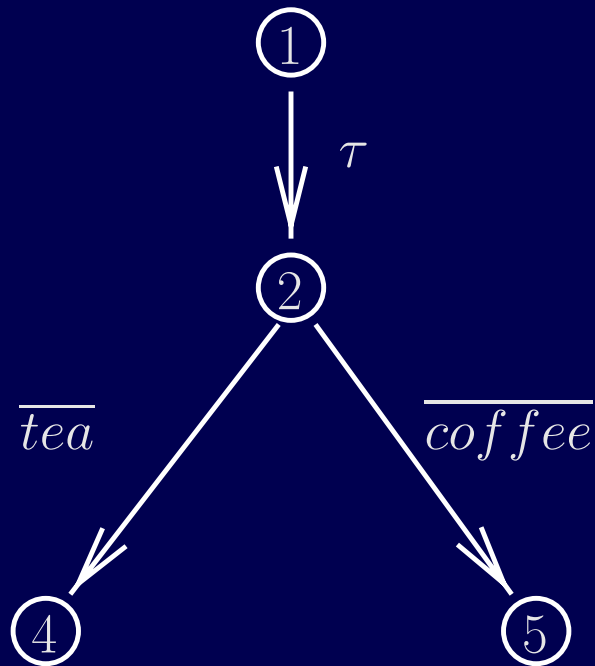
What can we do with the CWB?

- define agents
- simulate agents
- check equalities
- check properties from the modal μ -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



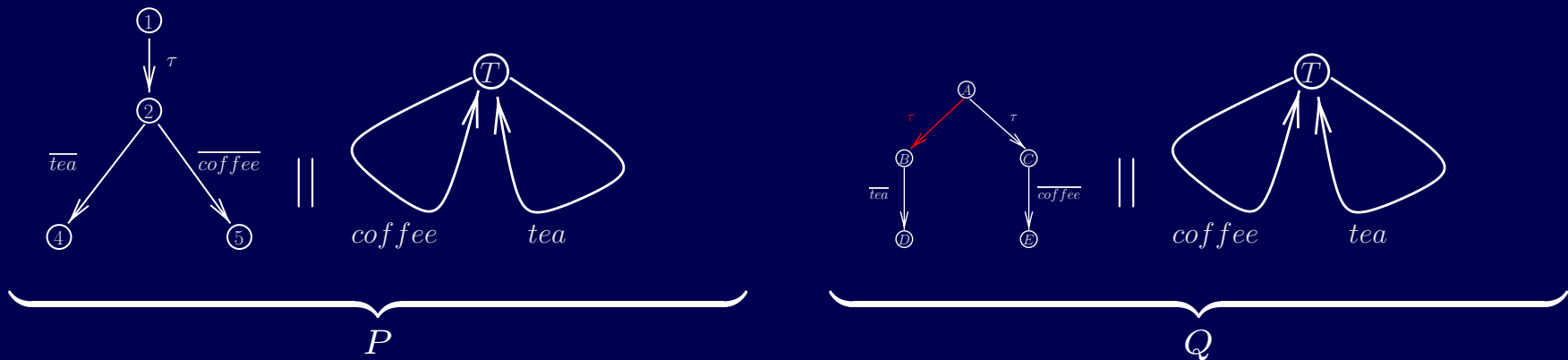
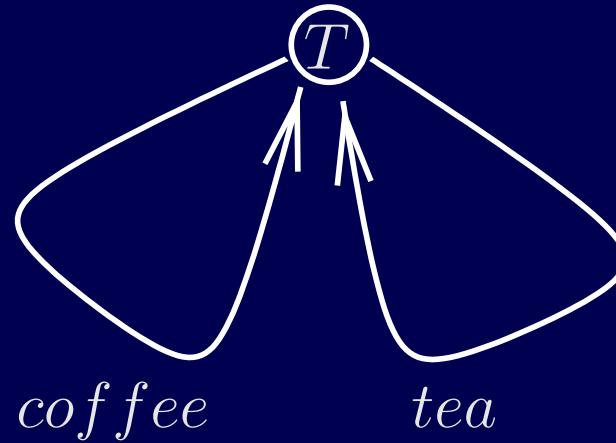
not bisimilar

not weakly bisimilar

} cannot match $A \xrightarrow{\tau} B$

Drinking Coffee

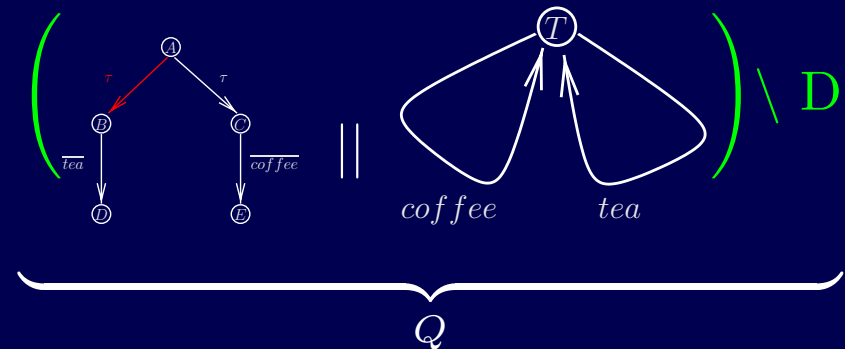
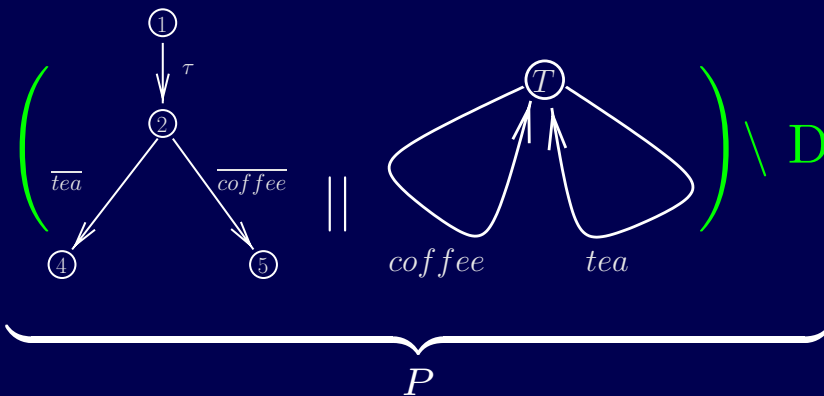
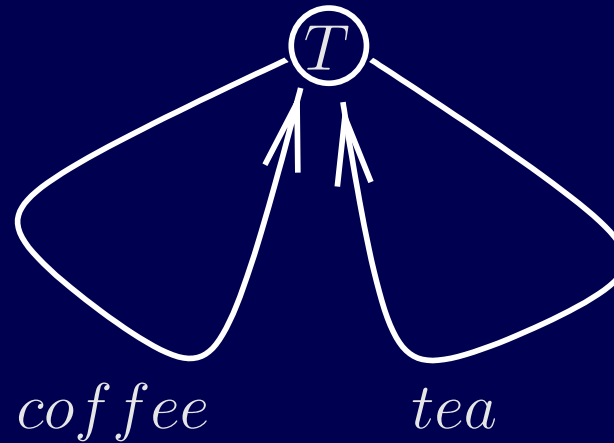
Add a consumer:



Now P and Q are even strongly bisimilar

Drinking Coffee

Add a consumer:



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

$$D := \{tea, coffee\}$$

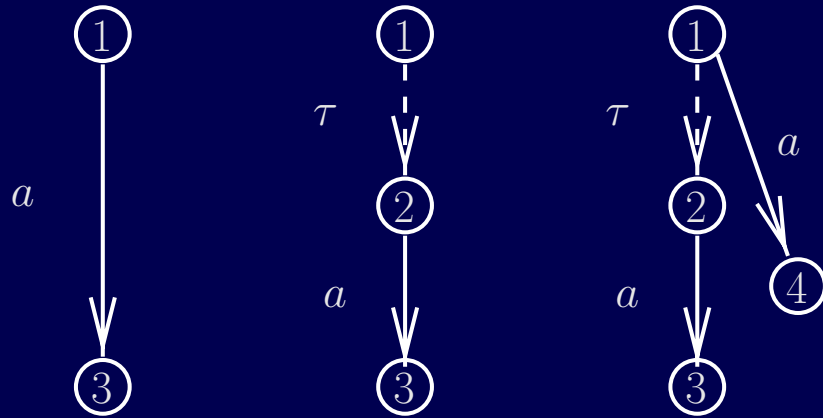
Reminder: Levels of Equivalence

\sim **strongeq** congruence : $P \sim Q \Rightarrow P + R \sim Q + R$
 $P \parallel R \sim Q \parallel R$

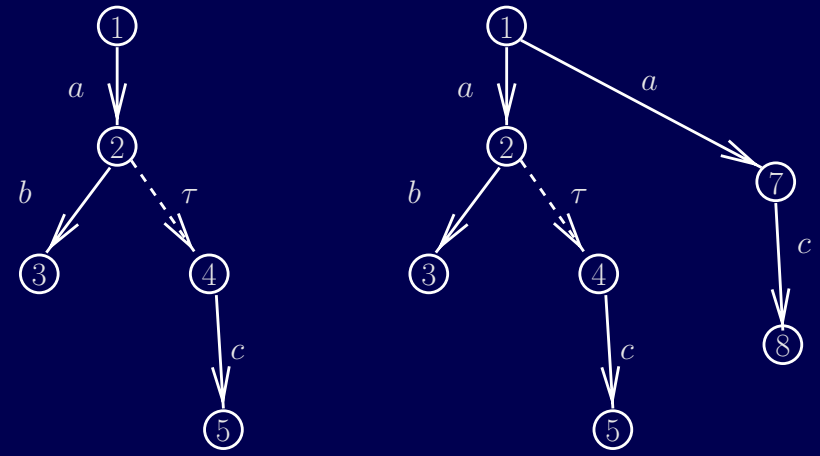
\approx **eq** process congruence : $P \approx Q \Rightarrow \alpha.P + R \approx \alpha.Q + R$
 $P \parallel R \approx Q \parallel R$

mayeq trace equivalence P can produce trace $\alpha \Leftrightarrow$
 Q can produce trace α

Additional (weak) Equalities

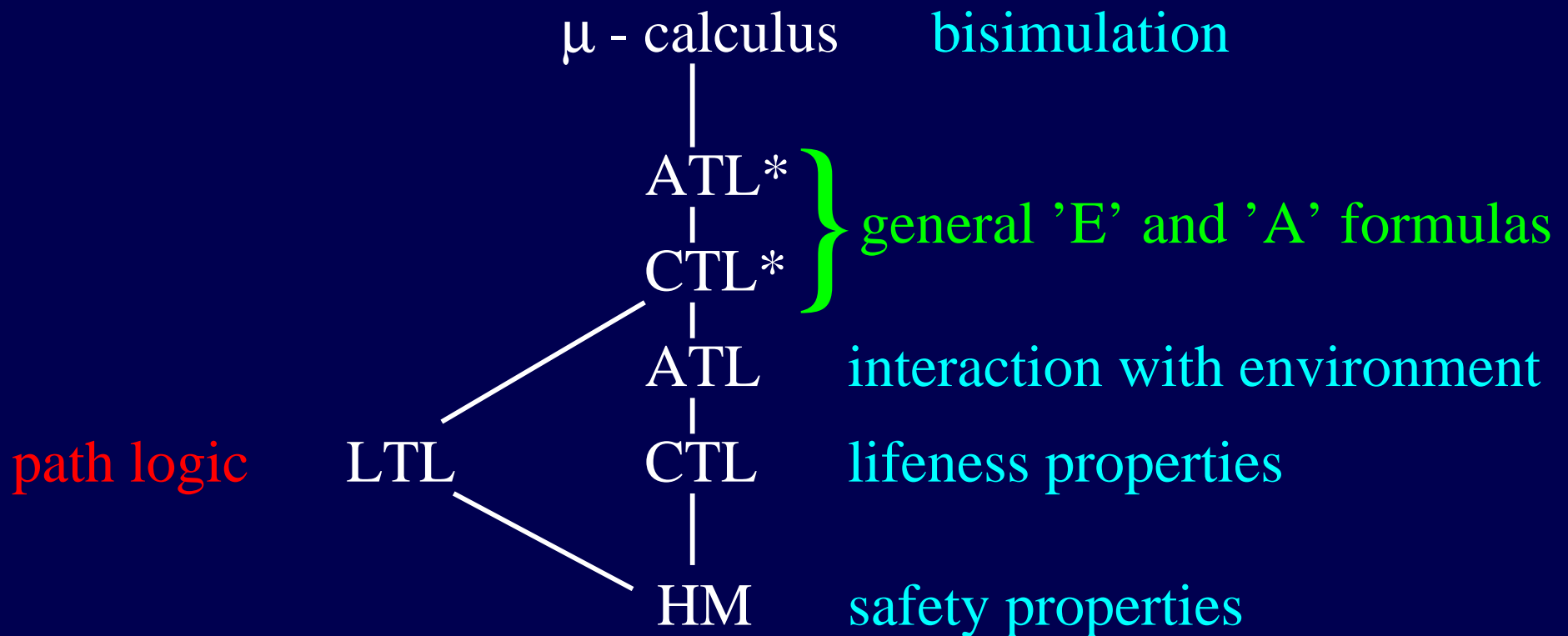


$$a \approx \tau.a \approx a + \tau.a$$



$$a.(b + \tau.c) \approx a.(b + \tau.c) + a.c$$

Temporal Logics



μ Calculus Syntax in CWB

PROP	P	T	true
		F	false
		$\sim P$	negation
		$P \ \& \ P$	conjunction
		$P \ \ P$	disjunction
		$P \Rightarrow P$	implication
		$[a, \dots]P$	strong necessity
		$[-a, \dots]P$	strong complement necessity
		$[[a, \dots]]P$	weak necessity
		$\langle a, \dots \rangle P$	strong possibility
		$\langle\langle a, \dots \rangle\rangle$	weak possibility

Special non-labels

tau: unobservable action

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

Special non-labels

tau: unobservable action

- $\text{tau.a.0} \models \langle \text{tau} \rangle \langle a \rangle T$
- $\text{tau.a.0} \not\models \langle a \rangle T$
- $\text{tau.tau.a.0} \not\models \langle \text{tau} \rangle \langle a \rangle T$
- $\text{tau.a.0} \models \langle \langle \text{tau} \rangle \rangle \langle a \rangle T$ not allowed

eps: empty observation

- $\text{tau.a.0} \models \langle \langle \text{eps} \rangle \rangle \langle a \rangle T$
- $\text{a.0} \models \langle \langle \text{eps} \rangle \rangle \langle a \rangle T$
- $\text{a.b.0} \models \langle -\text{eps} \rangle \langle b \rangle T$

Fixpoint Operators

$\min(X.P)$ least fixpoint temporal formula

$\max(X.P)$ greatest fixpoint temporal formula

$\max(Z.\varphi \ \& \ [-]Z)$	$AG \ \varphi$	Invariant
$\max(Z.[a]F \ \& \ [-]Z)$	$AG \ [a]F$	Safety: Never a
$\min(Z.<a>T \ \ <->Z)$	$EF \ <a>T$	Eventually a
$\min(Z.[-a]F \ \ (<->T \ \& \ [-]Z))$	$AF \ (<a>T \ \& \ [-a]F)$	Inevitably a
$\min(Z.Q \ \ (P \ \& \ <->T \ \& \ [-]Z))$	$P \ \text{Until} \ Q$	strong until
$\max(Z.Q \ \ (P \ \& \ [-]Z))$	$P \ \text{Wuntil} \ Q$	weak until
$\max(Z.[a]\min(Y.<->T \ \& \ [-b]Y) \ \& \ [-]Z)$	$AG(a \ \Rightarrow \ AFT)$	Response

What we have *not*:

- a notion of **states** or local propositions
- a global **store**
- an easy way to check, that a μ -formula and **intuition** coincide

Common Pitfalls

tau and eps

true: $\langle\langle\text{eps}\rangle\rangle \equiv \langle\langle\text{-S}\rangle\rangle \equiv \langle\text{tau}\rangle^*$

false: $\langle\langle\text{-eps}\rangle\rangle \equiv \langle\langle\text{S}\rangle\rangle$

e.g. $\text{tau.a.0} \models \langle\langle\text{-eps}\rangle\rangle\langle\text{a}\rangle\text{T}$ but $\text{tau.a.0} \not\models \langle\langle\text{S}\rangle\rangle\langle\text{a}\rangle\text{T}$

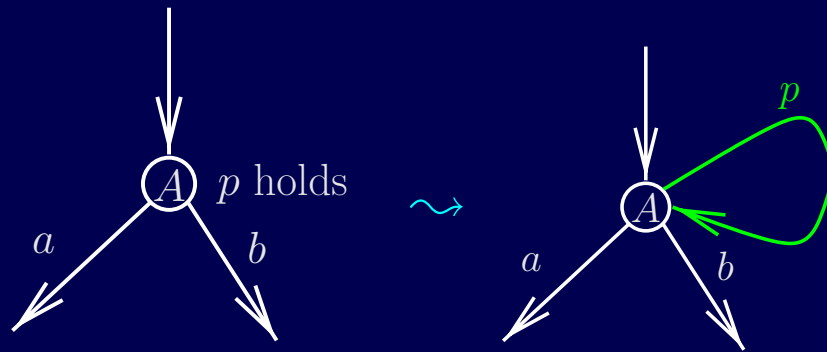
Modalities in fixed points

$\max(\text{Z}.\varphi \ \& \ [\text{-}]\text{Z})$: φ always

$\max(\text{Z}.\varphi \ \& \ [\text{S}]\text{Z})$: φ in all paths **excluding tau**

S: set of all observable actions

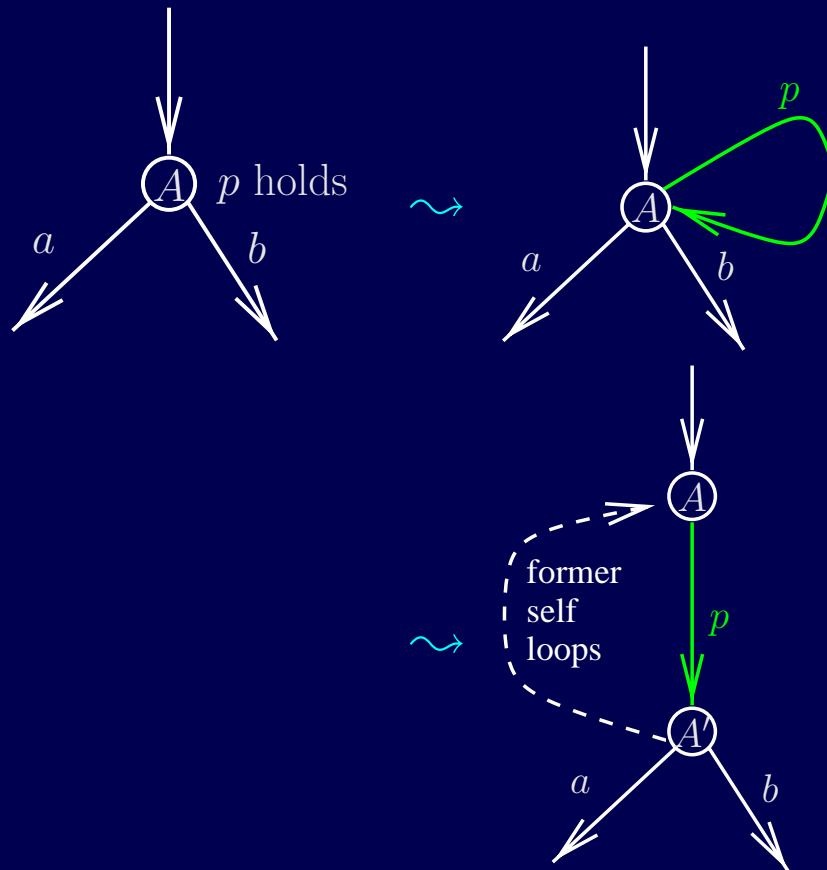
Living without Propositional Formulas



Problems:

- deadlock properties not preserved
- AF properties fail now

Living without Propositional Formulas



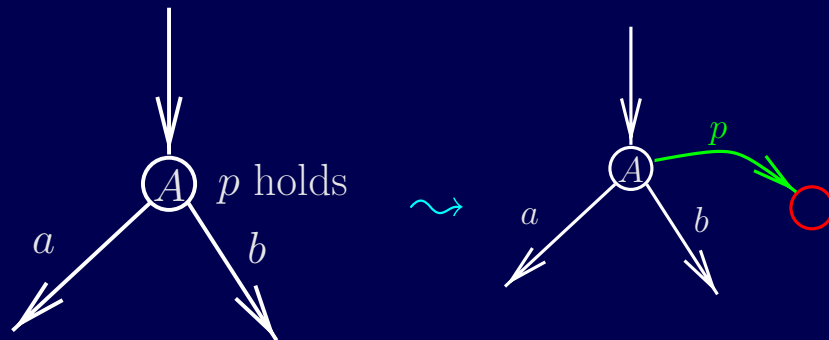
Problems:

- 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:

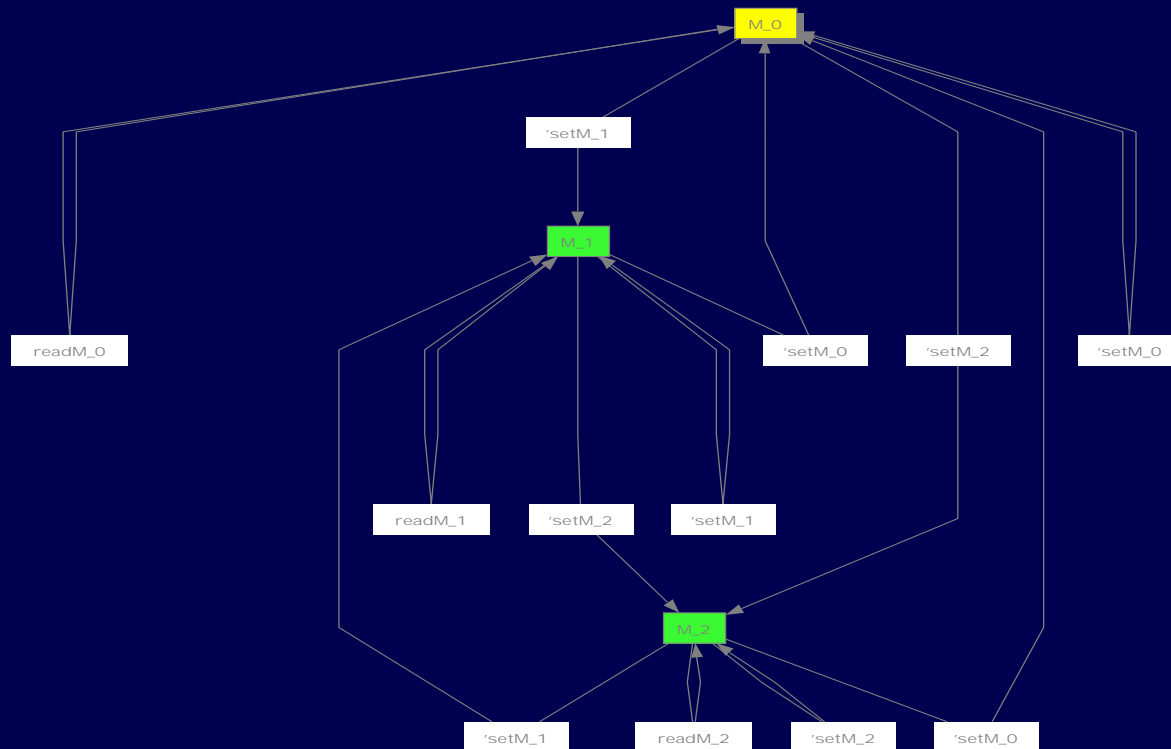
- introduces deadlocks
- AF properties fail now

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

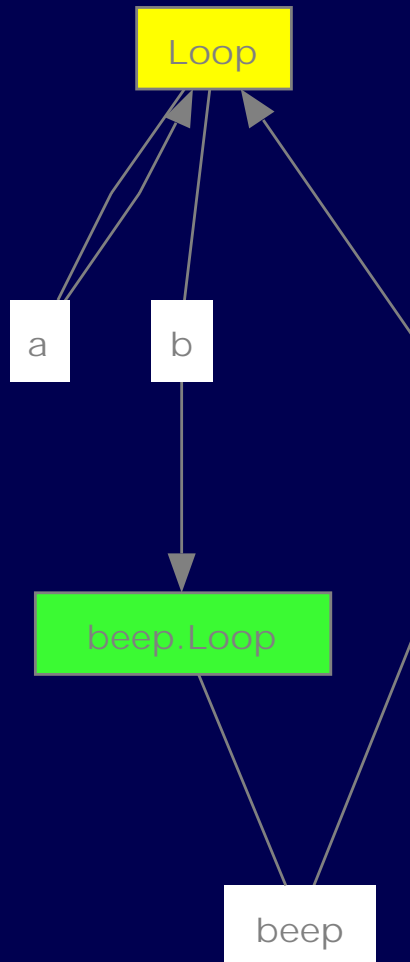
Variable M of type `int [0..2]`



Problems:

- Sequential Queries
- Tedious

Weak Fairness



Want:

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

1. Attempt:

Always, b will eventually be taken

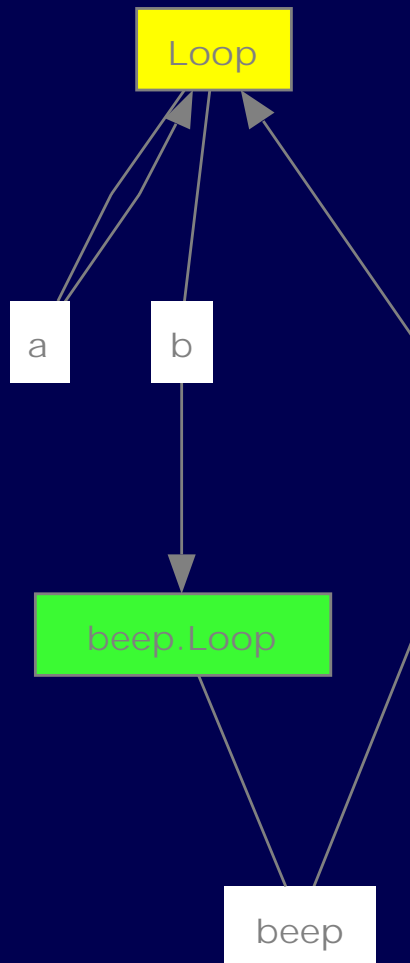
$$\nu \mathbf{X} . \mu \mathbf{Y} . (\langle - \rangle \mathbf{T} \wedge [-\mathbf{b}] \mathbf{Y}) \wedge [-] \mathbf{X}$$

2. Attempt:

If a is taken ∞ often, then so is b

$$\mu \mathbf{X} . \nu \mathbf{Y} . (\langle a \rangle \mathbf{T} \vee \mathbf{X}) \wedge [-\mathbf{b}] \mathbf{Y}$$

Weak Fairness



Want:

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

1. Attempt:

Always, b will eventually be taken

$$\nu \mathbf{X} . \mu \mathbf{Y} . (\langle - \rangle \mathbf{T} \wedge [-\mathbf{b}] \mathbf{Y}) \wedge [-] \mathbf{X}$$

2. Attempt:

If a is taken ∞ often, then so is b

$$\mu \mathbf{X} . \nu \mathbf{Y} . (\langle a \rangle \mathbf{T} \vee \mathbf{X}) \wedge [-\mathbf{b}] \mathbf{Y}$$

Problem:

We can *express* fairness,

but not add it as an *Assumption*

‘inevitably, it will beep’ is equivalent to **false**.

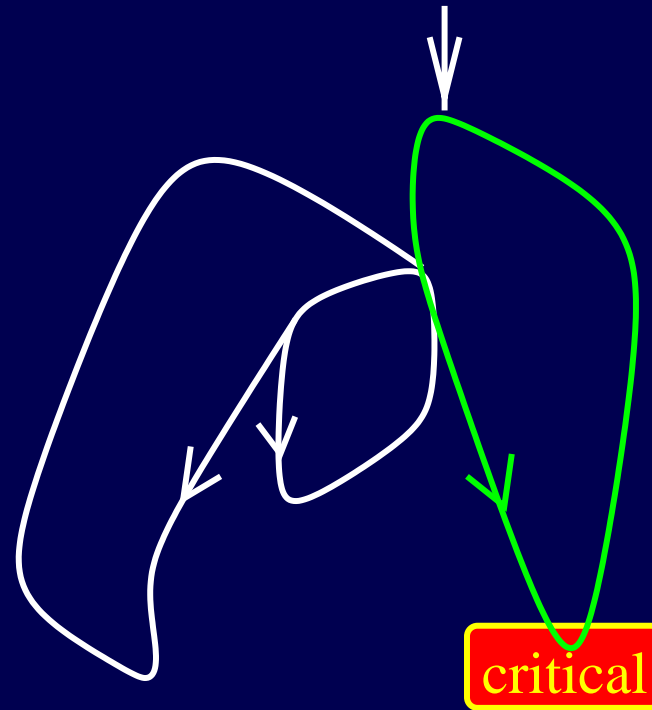
AND: our formulas are equivalent to **false**.

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
```

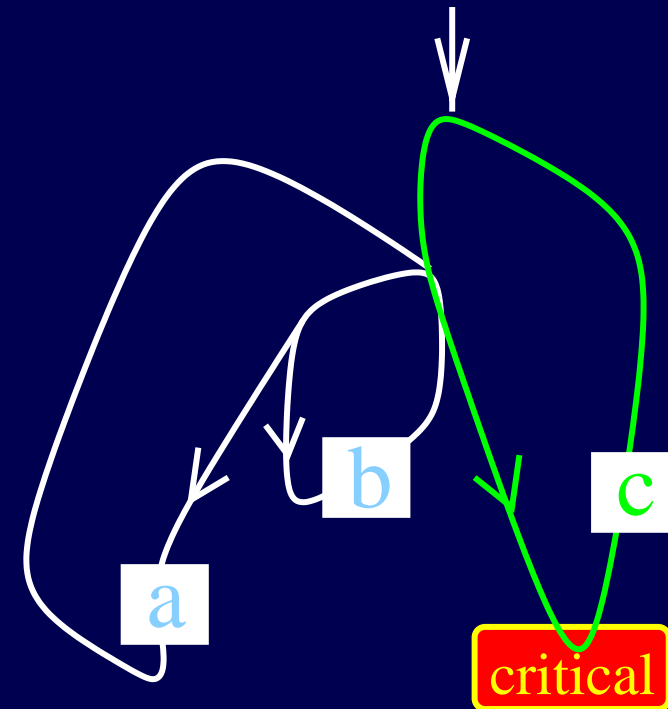
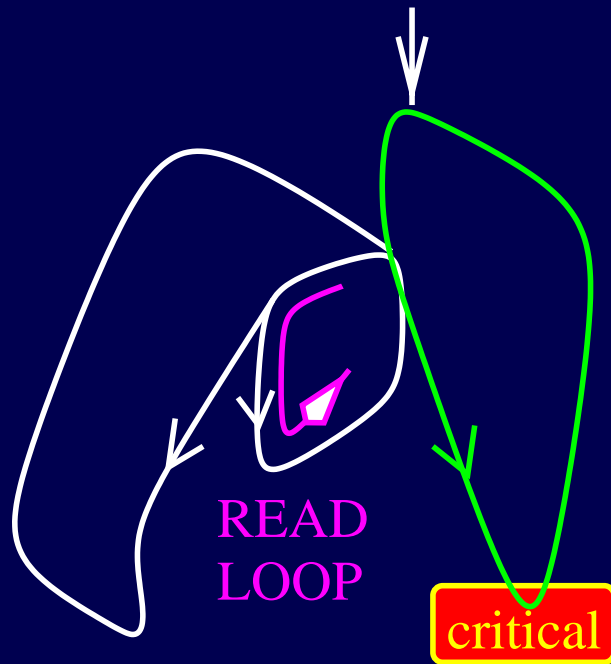

Dekker's Mutex Algorithm

```
**** 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:

$$\text{System} \models \nu Z. \langle - \rangle T \wedge [-] Z$$

2. It is impossible to reach fair loops:

$$\forall \text{actions } x : \text{System} \not\models \mu Z. (\nu X. [[-x]] F \wedge [[x]] X) \vee \langle - \rangle Z$$

3. If actions x , y happen ∞ often, then c happens ∞ often :

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

strong fairness $\xrightarrow{1,2}$ at least two actions are observed ∞ often
 $\xrightarrow{3}$ freedom of individual starvation

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 μ -calculus formulas
- experiments with own
 - process algebras
 - logics
 - modelchecking algorithms

Limitations

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

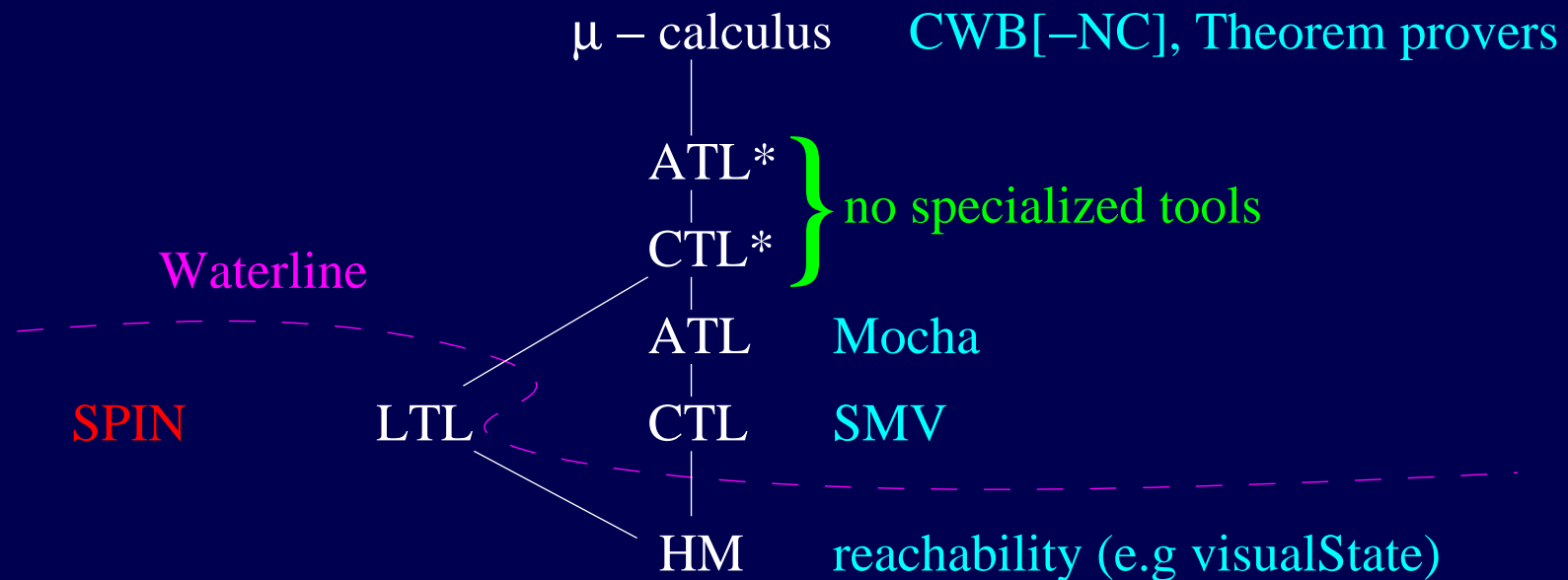


technologies that require experts



hands off: non-proven technologies

Tools in Practice



Incomplete Methods: Simulation, (automated) Testing

Invitation: Dig Deeper

The user manual [The Edinburgh 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>