SUMMERSOC 2014
Wed July 3rd 10:30 - 12
Wed July 3rd 15 – 16.30

Tutorial Formal Methods for SOC 2. Temporal Logic and Model Checking

Wolfgang Reisig

Humboldt Universität Informatik

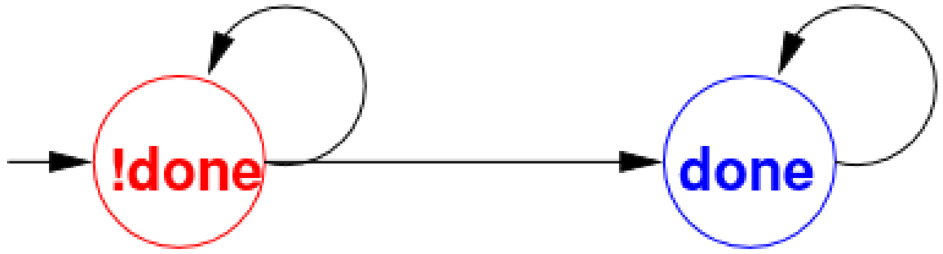
Theory of Programming

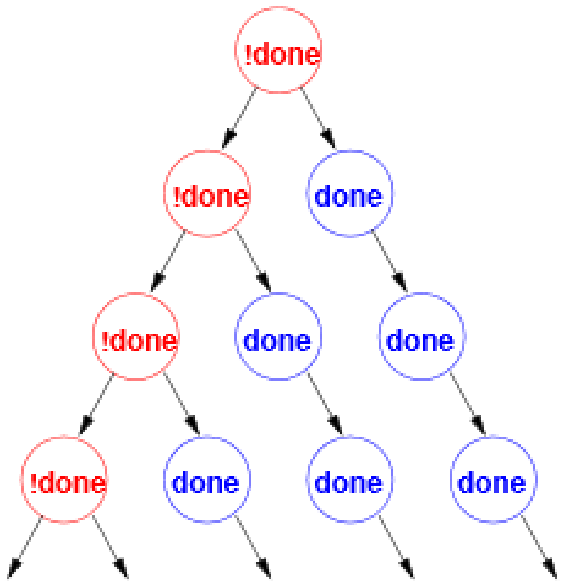
Prof. Dr. W. Reisig

1. Temporal Logic

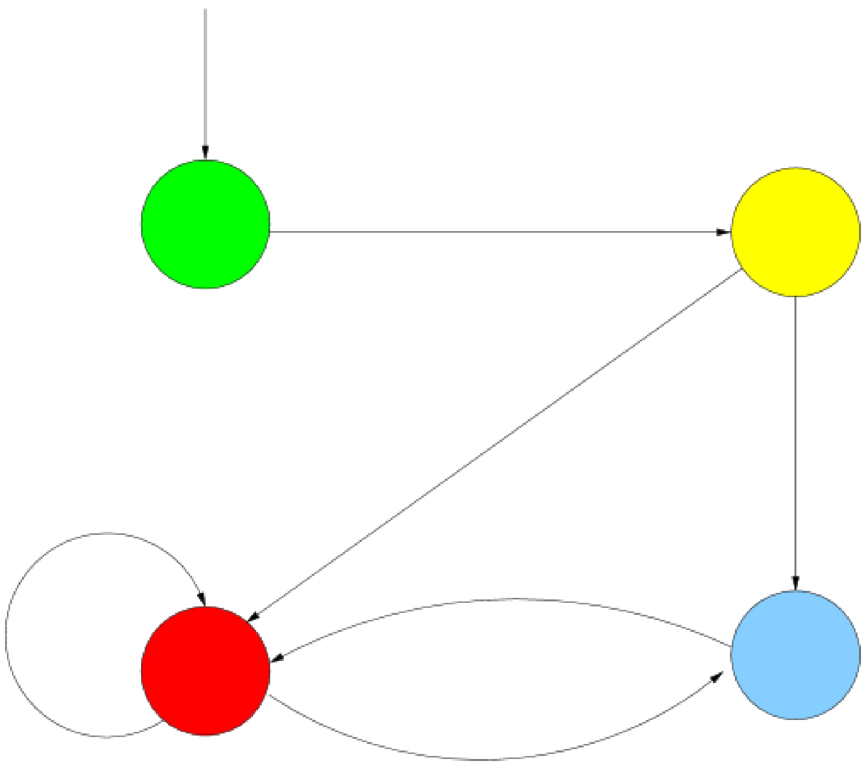
How to express properties
of systems that perform discrete steps?

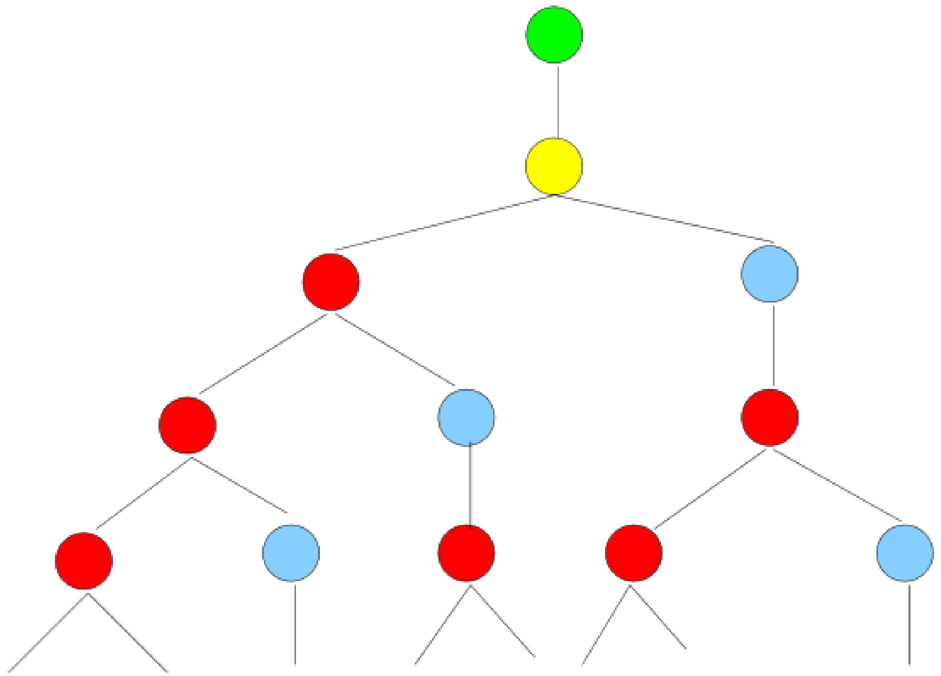
From a transition system to its tree



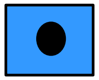


Once more: a process and its tree





Computation Tree Logic *CTL**

p = 

eventually p

globally p

next p

p until q

Computation Tree Logic *CTL**

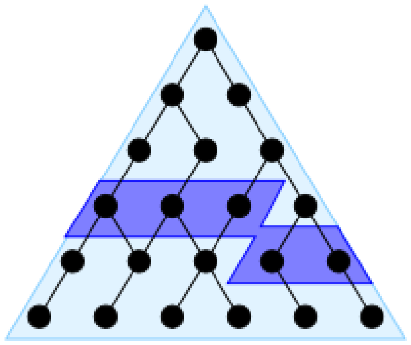
p = ●

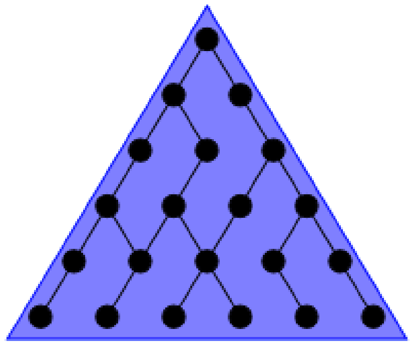
eventually p

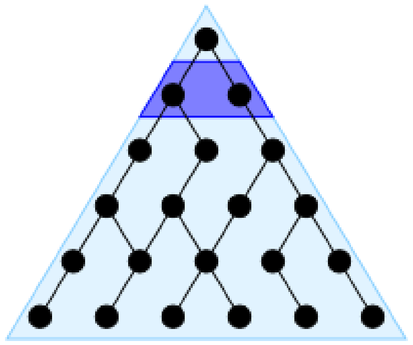
globally p

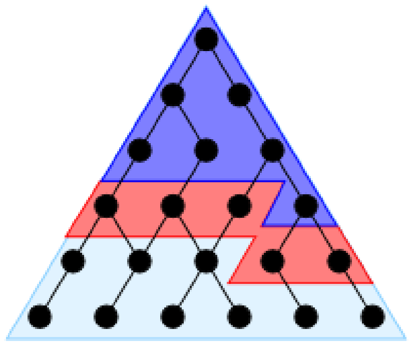
next p

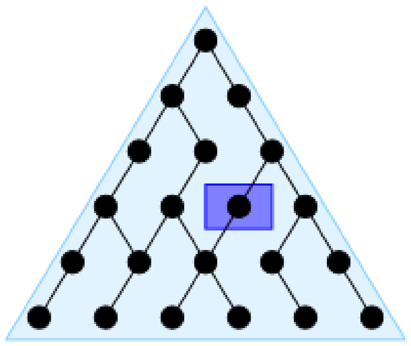
p until q

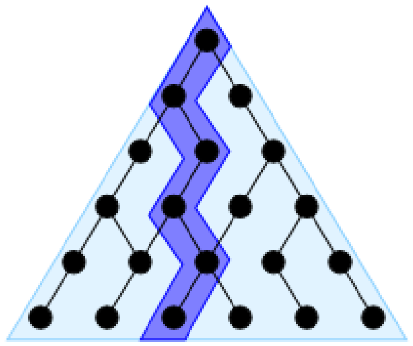

AF p

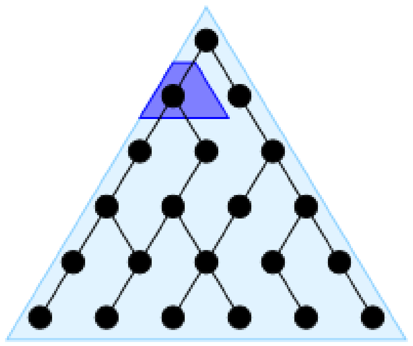

AG p

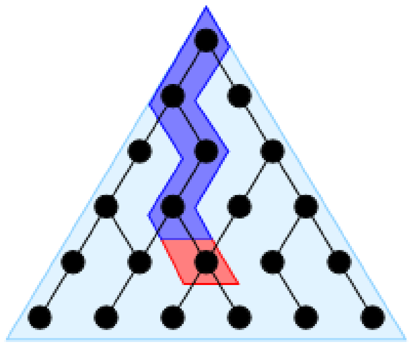

AX p


A [p U q]

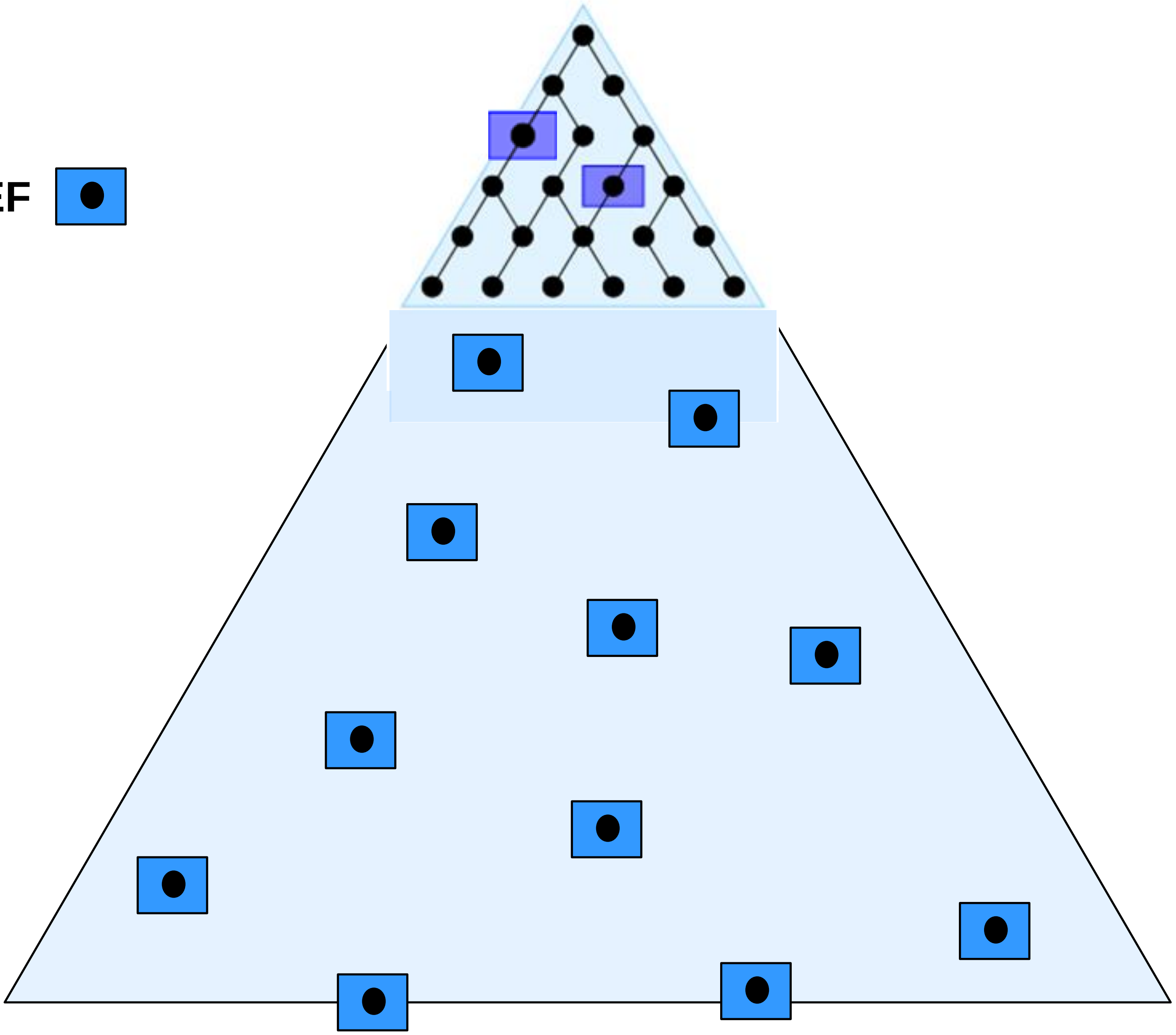

EF p


EG p

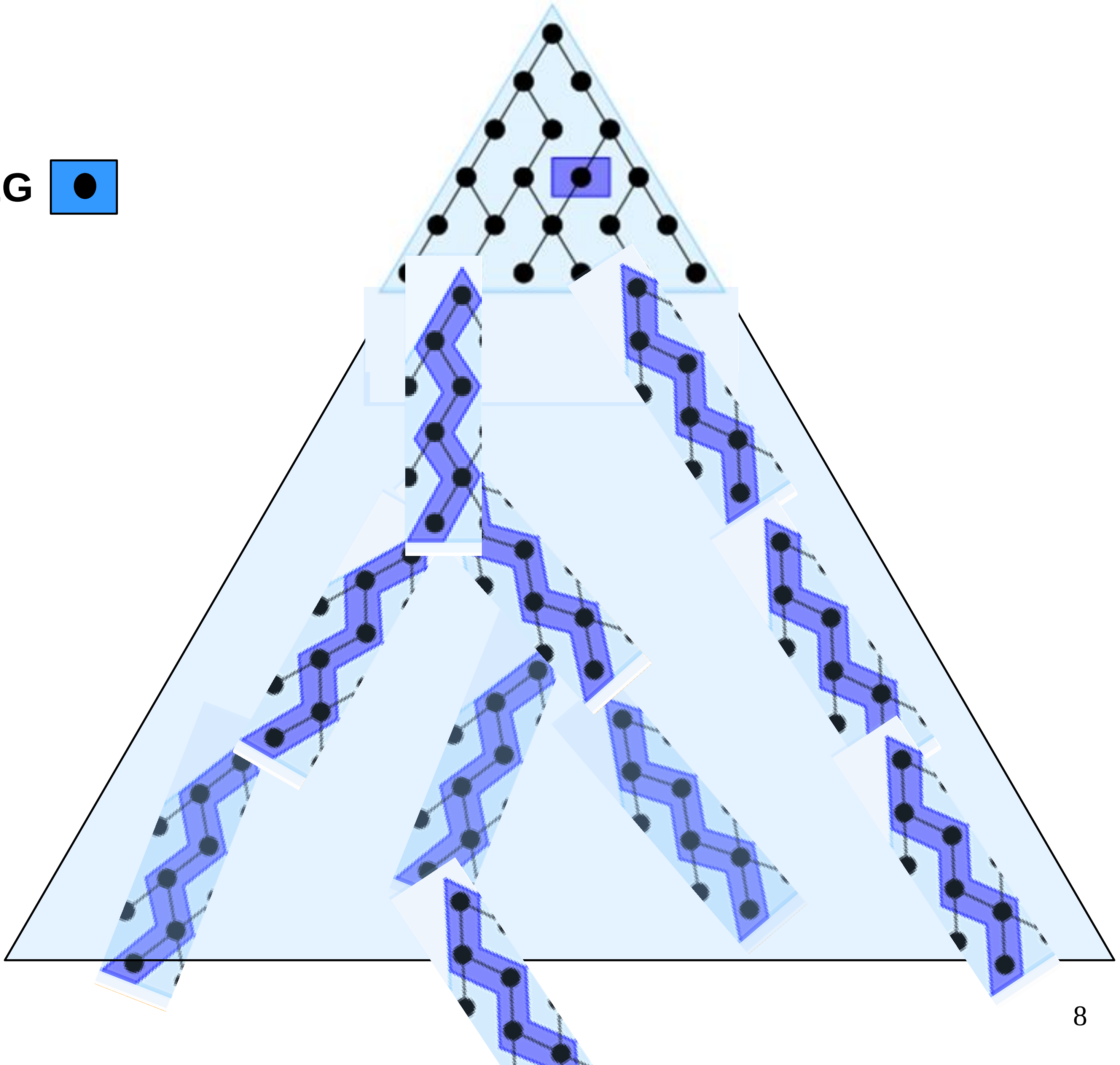

EX p


E [p U q]

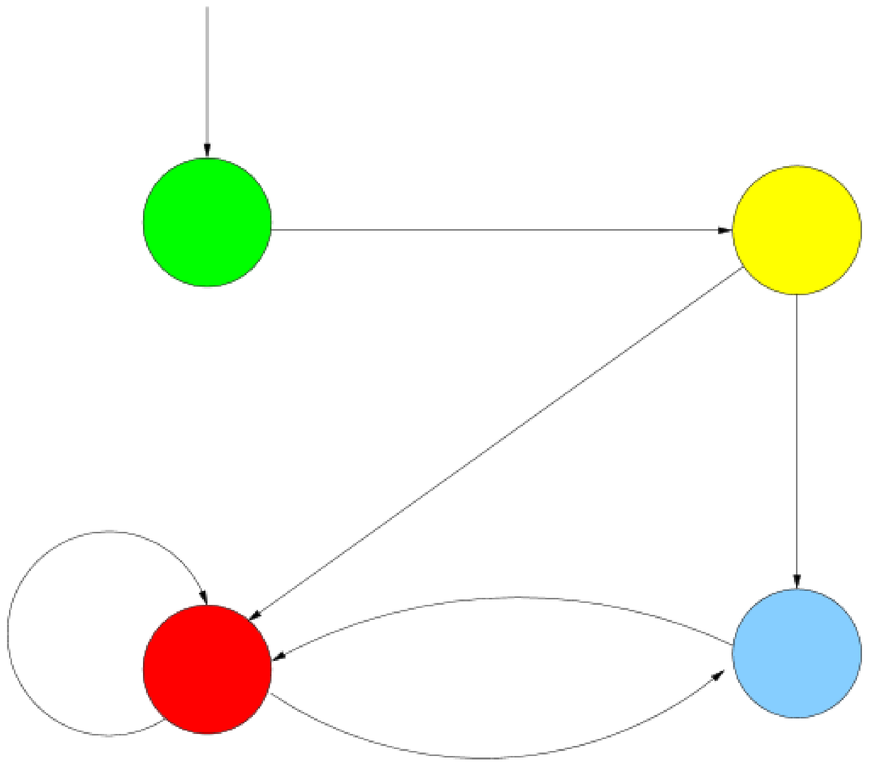
AGEF ●

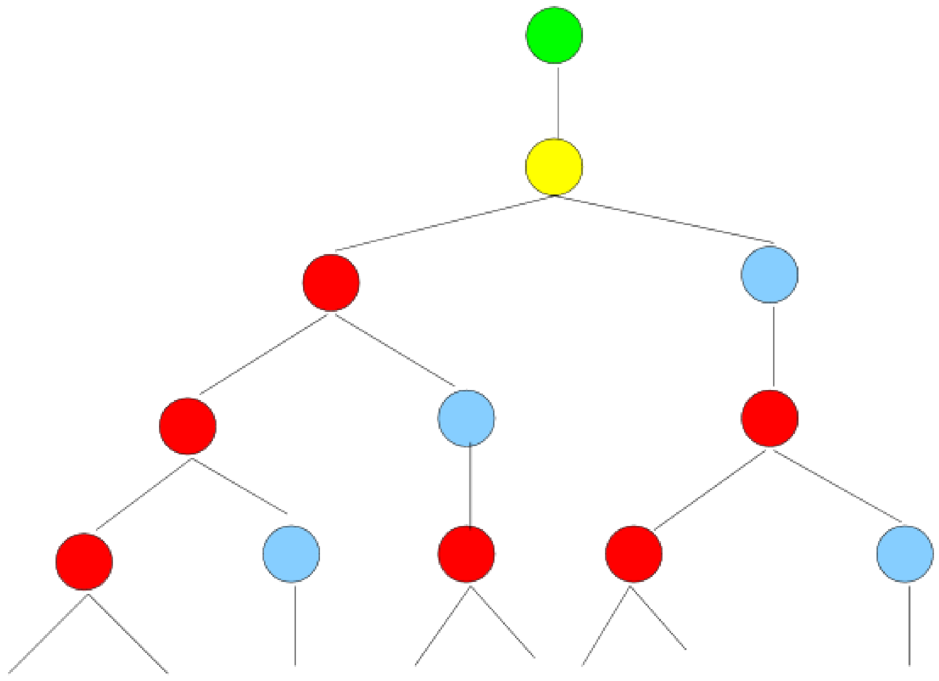


AGEG ●



Valid formulas





🟢

EX 🟡

AX 🟡

AG (🟢 ∨ EX 🔴)

AGEF 🔵

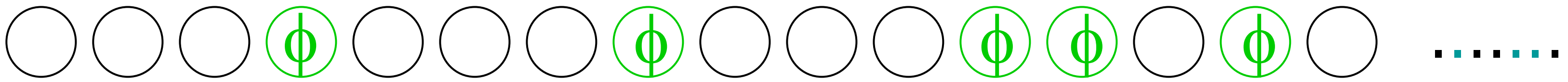
EFG 🔴

Typical applications

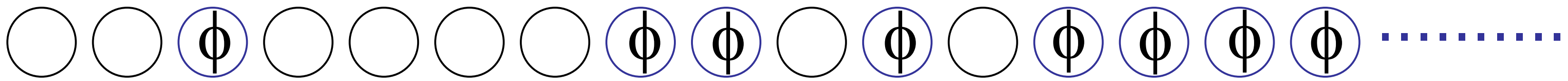
“Never something bad happens”	AG *safely*
“No deadlock reachable”	AG *enabled*
„With a series of clicks you reach p“	EF *p*
“Whatever happens – you will succeed”	AF *Goal*
“Each requirement is followed by an acknowledgement”	AG(*req* U AF *ack*)
“It makes sense to wait”	AG AF *avail*
“You always can properly terminate”	AG EF *exit*

formulas interpreted in paths

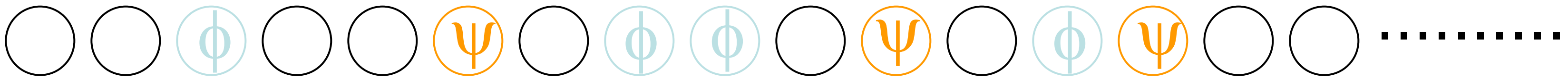
$G\,F\,\phi$ = ϕ holds infinitely often



$F\,G\,\phi$ = ϕ stabilizes



$G\,(\phi \blacklozenge F\,\psi)$ = ϕ leads to ψ



Tautologies: $F\,G\,F\,\phi \blacklozenge G\,F\,\phi$ $G\,F\,G\,\phi \blacklozenge F\,G\,\phi$

Why not just First order logic (predicate logic)?

Example:
Whenever process A sends a message to process B, then B eventually sends an acknowledgement to A.

First order:
🕕 t (send(A,B,t) ♏ 🕖 t’ (greater(t’,t) 📬 send(B,A,t’)))

CTL*:
AG (Send (A,B) ♏ AF Send (B,A))

Expressiveness

Why just ***THIS*** logic?

Theorem. next lecture

Two states are bisimilar

iff they share the same *CTL** properties.

Consequence:

Specify a system in terms of CTL*.

This may yield various different implementations.

They all are bisimular.

2. Model Checking

How to verify properties
of systems that perform discrete steps?

Why verify a system design?

to prove its correctness (theoretically)

To find subtle mistakes (practically)

In contrast: Testing
Testing shows presence of mistakes,
but not their absence (E. Dijkstra)

Verification techniques

manually	Hoare Logic, Invariants, descending functions, ...
interactive	Theorem proving
automatically	Model Checking

Model Checking

Aim: Show that a CTL* formula ϕ holds in a transition system T .

Idea: Visit each state of T and derive its properties.
Combine the results to prove ϕ

First relevant results: 1986

Brake through: 1992

… a success story
with a fundamental problem:
state explosion

State Explosion

Assume: 2.4 GHz, sufficient store,
one new state per clock cycle:
how many states can you visit?

2,400,000,000 per second

144,000,000,000 per minute

8,840,000,000,000 per hour

207,360,000,000,000 per day

75,738,240,000,000,000 per year

1,514,764,800,000,000,000,000,000,000 since big bang
($< 10^{28}$)

Systems with 10^{28} states

Theoretically: 90 Boolean variables

Practically: 200 Boolean variables (in distributed systems)

Milestones of Model Checking:

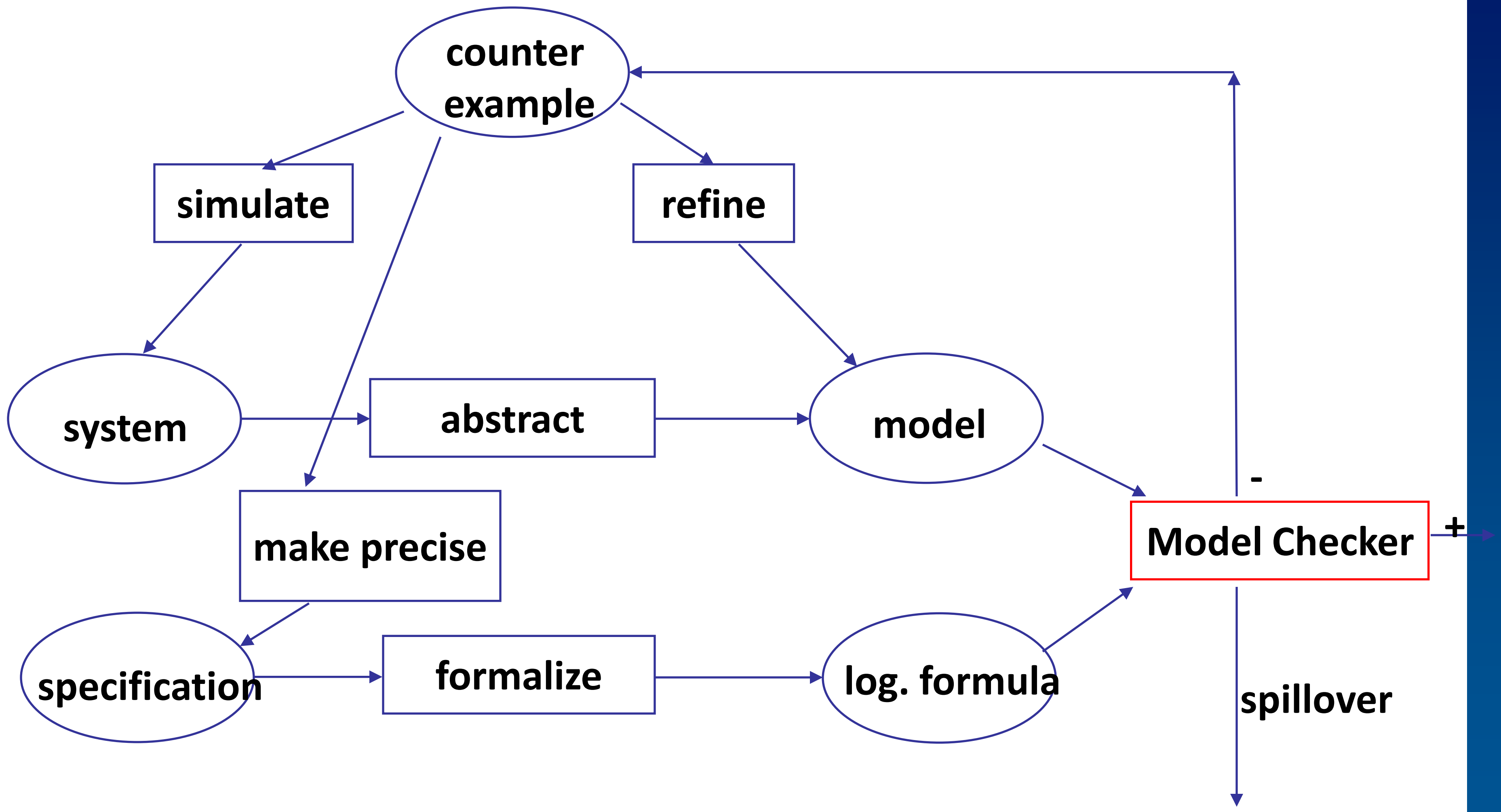
1986: 10^{6}
1992: 10^{20}
1996: 10^{100}
2000: 10^{1000}

A miracle?
Cheating?
Clever technolgy?

Supporting techniques:

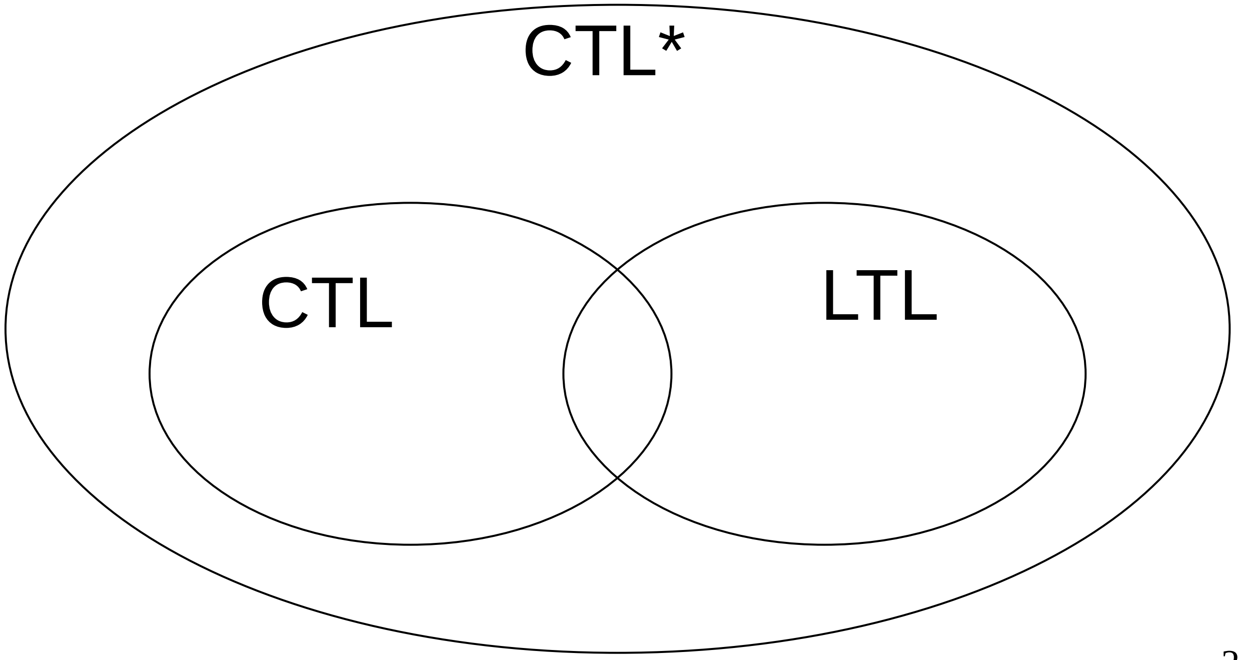
Abstract interpretation,
Symbolic Model checking.

Model Checking: How to use it



Efficient algorithms

… not for CTL*,
but for subsets of it



Path Formula: may hold in an path

proposition p
p ” (s_0 s_1 s_2 s_3 ...) iff p ” s_0

X *path formula*
X ϕ ” (s_0 s_1 s_2 s_3 ...) iff ϕ ” (s_1 s_2 s_3 ...)

F *path formula*
F ϕ ” (s_0 s_1 s_2 s_3 ...) iff ϕ ” (s_i s_{i+1} s_{i+2} ...) for some i

G *path formula*
G ϕ ” (s_0 s_1 s_2 s_3 ...) iff ϕ ” (s_i s_{i+1} s_{i+2} ...) for all i

path formula U *path formula*
ϕ U ψ ” (s_0 s_1 s_2 s_3 ...) iff ...

State Formula:
may hold in a state of a tree

E *path formula*
E ϕ ” s iff for some path π starting at s holds: ϕ ” π

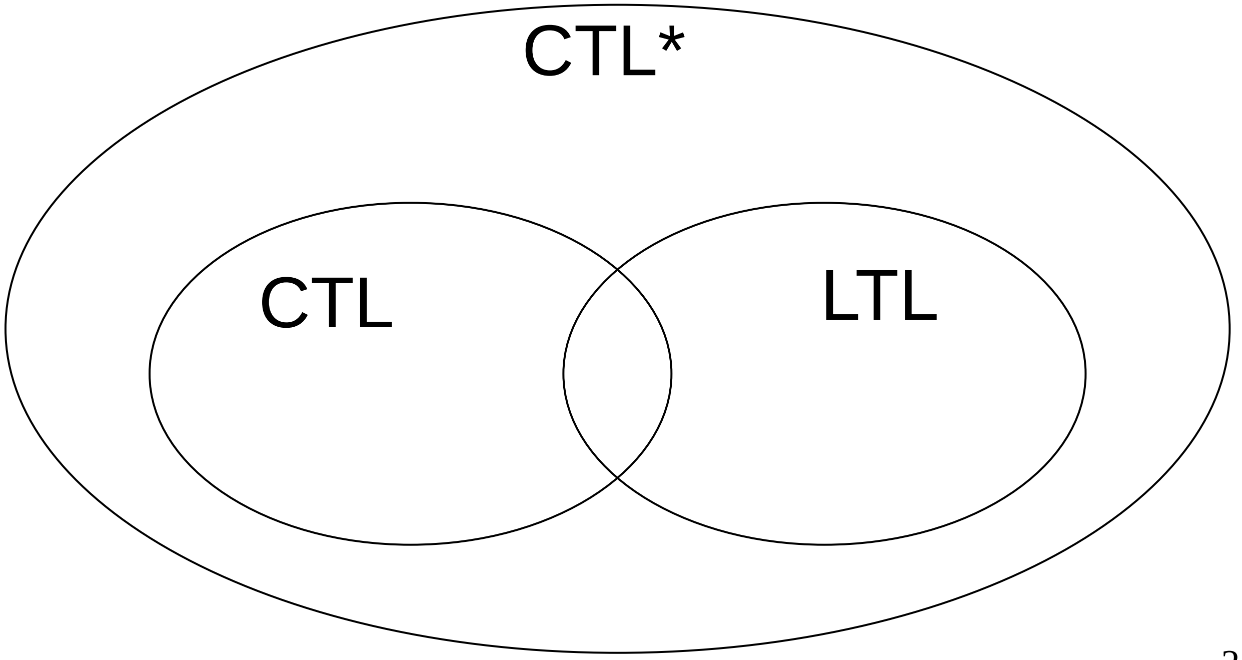
A *path formula*
A ϕ ” s iff for each path π starting at s holds: ϕ ” π

Efficient algorithms

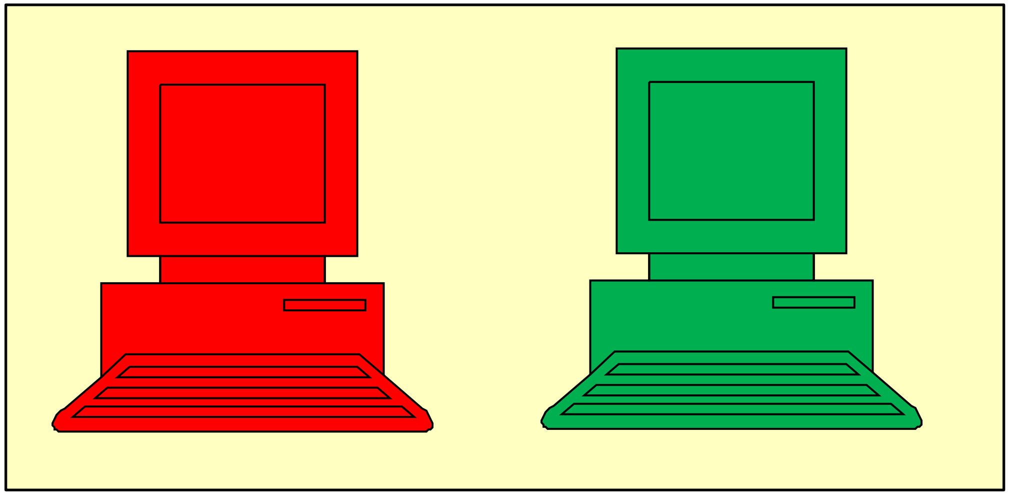
CTL* : $O(2^{|\phi|}\ |TS|)$

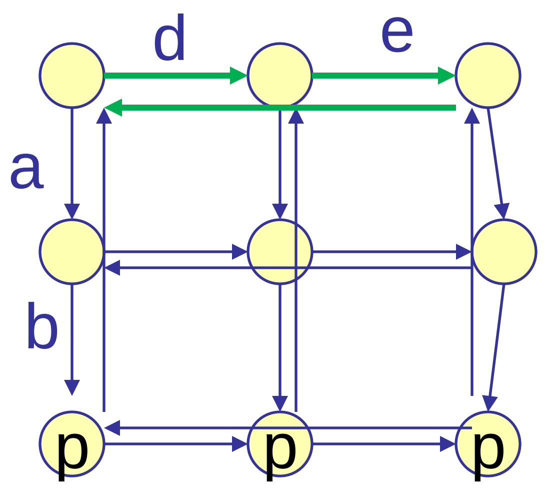
LTL: Only path formulas : $O(2^{|\phi|}\ |TS|)$

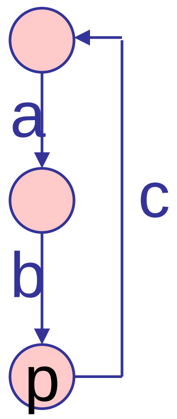
CTL: Only state formulas: $O(|\phi|\ |TS|)$

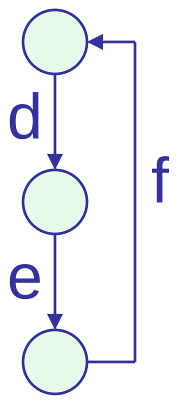


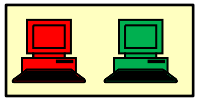
Fairness



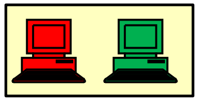






 ✝ GFp

” GFp

Take ***GFp*** as part of the specification of 

SUMMERSOC 2014
Wed July 3rd 10:30 - 12
Wed July 3rd 15 – 16.30

Tutorial Formal Methods for SOC 2. Temporal Logic and Model Checking

Wolfgang Reisig

the end

Humboldt Universität
Informatik

Theory of Programming

Prof. Dr. W. Reisig