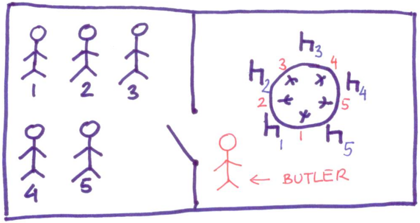
Solutions to Some Problems

SE 3BB4

Ryszard Janicki

Department of Computing and Software, McMaster University, Hamilton, Ontario, Canada

- **No more than 4 philosophers are sitting at the table.**



$FORK = (get \rightarrow put \rightarrow FORK)$

$PHIL = (think \rightarrow sitdown \rightarrow right.get \rightarrow left.get \rightarrow eat \rightarrow$
$\quad right.put \rightarrow left.put \rightarrow getup \rightarrow PHIL)$

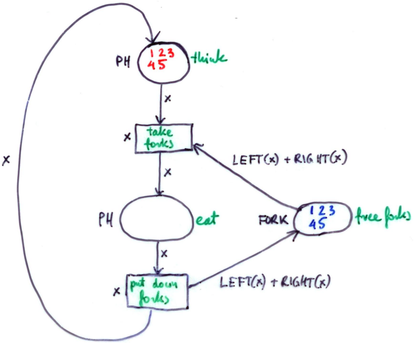
$BUTLER(K = 4) = COUNT[0]$

$COUNT[i : 1..4] = (when(i < K)\ sitdown \rightarrow COUNT[i+1] \mid$
$\quad getup \rightarrow COUNT[i-1]$

$\| DINERS(N = 5) = (forall[i : 1..N]$
$\quad (phil[i] : PHIL \parallel \{phil[i].right, phil[i \oplus 1].left\} :: FORK)$
$\quad \parallel \underbrace{\{phil[i : ..N]\}}_{\{phil[1], phil[2], phil[3], phil[4], phil[5]\}} :: BUTLER(K = 4))$

'Butler' Solution

- 'Butler' solution works. No deadlock and no starvation.
- *FORK*'s are *passive* processes (monitors), hence they always can be presented as:
  $FORK = (get \rightarrow put \rightarrow FORK)$
- *PHILOSOPHER*'s are *active* processes.



$colour\ PH = with\ ph1 \mid ph2 \mid ph3 \mid ph4 \mid ph5$

$colour\ Fork = with\ f1 \mid f2 \mid f3 \mid f4 \mid f5$

$LEFT : PH \rightarrow FORK, \quad RIGHT : PH \rightarrow FORK$

$var\ x : \ PH$

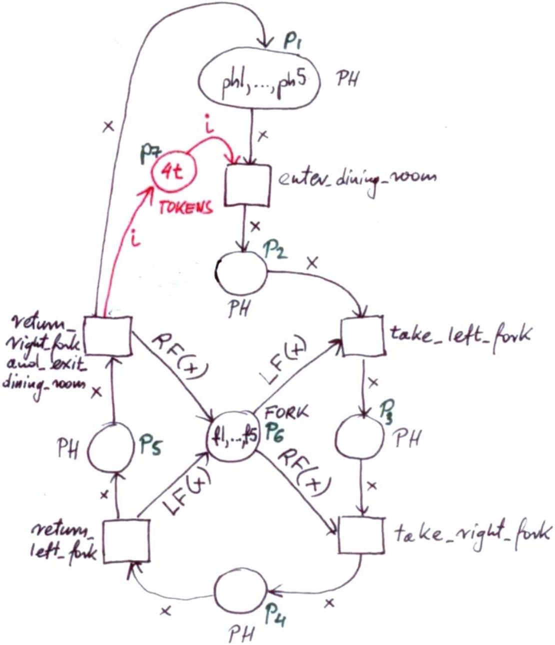
$fun\ LEFT\ x = case\ of\ ph1 \Rightarrow f2 \mid ph2 \Rightarrow f3 \mid ph3 \Rightarrow f4 \mid$
$ph4 \Rightarrow f5 \mid ph5 \Rightarrow f1$

$fun\ RIGHT\ x = case\ of\ ph1 \Rightarrow f1 \mid ph2 \Rightarrow f2 \mid ph3 \Rightarrow f3 \mid$
$ph4 \Rightarrow f4 \mid ph5 \Rightarrow f5$

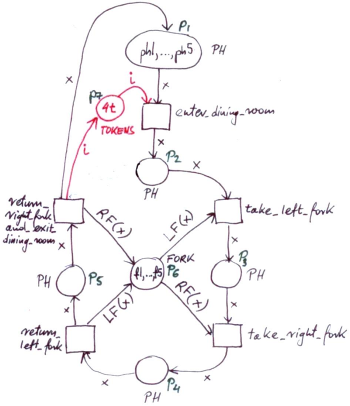
- **Provide a Coloured Petri Net solution to Dining Philosophers with a butler. Prove that this solution is deadlock-free.**

Solution:

colour PH = with ph1 | ph2 | ph3 | ph4 | ph
colour FORK = with f1 | f2 | f3 | f4 | f5
colour TOKENS = with t
var x : PH
var i: TOKENS
fun LF x = case of ph1 $\Rightarrow$ f2 | ph2 $\Rightarrow$ f3 | ph3 $\Rightarrow$ f4 | ph4 $\Rightarrow$ f5 | ph5 $\Rightarrow$ f1
fun RF x = case of ph1 $\Rightarrow$ f1 | ph2 $\Rightarrow$ f2 | ph3 $\Rightarrow$ f3 | ph4 $\Rightarrow$ f4 | ph5 $\Rightarrow$ f5



p1
ph1,...,ph5
PH
x
i
p7
4t
TOKENS
enter_dining_room
x
p2
PH
x
take_left_fork
return_right_fork and_exit dining_room
RF(x)
LF(x)
x
FORK
P6
f1,...,f5
P3
PH
PH
P5
x
RF(x)
LF(x)
return_left_fork
take_right_fork
x
x
P4
PH
x



Interpretation of places:

p1 - thinking room

p2 - philosophers without forks in the dining room

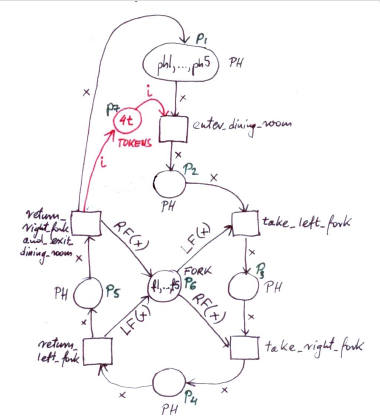
p3 - philosophers with left forks in the dining room

p4 - philosophers that are eating

p5 - philosophers that finished eating and still with right forks in the dining room

p6 - unused forks p7 - butler or counter

Invariants



inv1 $m(p1) + m(p2) + m(p3) + m(p4) + m(p5) =$
$ph1 + ph2 + ph3 + ph4 + ph5$

inv2 $|m(p7)| + |m(p2)| = 4$

inv3 $LF(m(p4)) + RF(m(p4)) + m(p6) = f1 + f2 + f3 + f4 + f4 + f5$

Proof

Now consider two cases:

1. $m(p4) + m(p5) \neq 0$. Then either *return_left_fork* or *return_right_fork_and_exit_dining_room* can be fired.
2. $m(p4) + m(p5) = 0$. Then from invariant [inv3] we have : $LF(m(p3)) + m(p6) = f1 + f2 + f3 + f4 + f4 + f5$ and from invariant [inv1]:
   $m(p1) + m(p2) + m(p3) = ph1 + ph2 + ph3 + ph4 + ph5$.
   From the definitions of $LF(x)$ and $RF(x)$ we have $LF(x) \neq RF(x)$ for all $x = ph1, ph2, ph3, ph4, ph5$. Hence if $m(p3) \neq 0$ then *take_right_fork* can be fired. Similarly if $m(p2) \neq 0$ then *take_left_fork* can be fired. If $m(p1) \neq ph1 + ph2 + ph3 + ph4 + ph5$, then either $m(p3) \neq 0$ or $m(p2) \neq 0$. If $m(p1) = ph1 + ph2 + ph3 + ph4 + ph5$ then $m(p2) = 0$, and from invariant [inv2] $|m(p7)| = 4$, so *enter_dining_room* can be fired.

'Hungry' and 'Asymmetrically Simple Minded', or 'Some Discipline Added'

- Philosophers 1, 3 and 5 always perform '*left.get* → *right.get*', while 2 and 4 always perform '*right.get* → *left.get*'.

$$FORK = (get \rightarrow put \rightarrow FORK)$$

$$\begin{aligned} PHIL = (&when(i = 1 \vee i = 3 \vee i = 5)\ think \rightarrow left.get \rightarrow \\ &right.get \rightarrow eat \rightarrow left.put \rightarrow right.put \rightarrow PHIL \\ &|\ when(i = 2 \vee i = 4)\ think \rightarrow right.get \rightarrow \\ &left.get \rightarrow eat \rightarrow right.put \rightarrow left.put \rightarrow PHIL) \end{aligned}$$

$$\| DINERS(N = 5) = forall[i : 1..N]$$
$$(phil[i] : PHIL \parallel \{phil[i].right, phil[i \oplus 1].left\} :: FORK)$$

- **Works! Neither deadlock nor starvation.**
- The Labelled Transition System is **very big**!

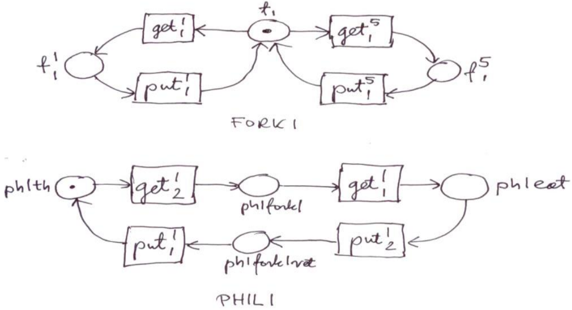
Asymmetrically Simple Minded Philosophers

- Notation: for get^i_j, put^i_j, i - philosopher number, j - fork number

$$FORK_1 = (get^1_1 \rightarrow put^1_1 \rightarrow FORK_1 \mid get^5_1 \rightarrow put^5_1 \rightarrow FORK_1)$$
$$FORK_2 = (get^2_2 \rightarrow put^2_2 \rightarrow FORK_2 \mid get^1_2 \rightarrow put^1_2 \rightarrow FORK_2)$$
$$FORK_3 = (get^3_3 \rightarrow put^3_3 \rightarrow FORK_3 \mid get^2_3 \rightarrow put^2_3 \rightarrow FORK_3)$$
$$FORK_4 = (get^4_4 \rightarrow put^4_4 \rightarrow FORK_4 \mid get^3_4 \rightarrow put^3_4 \rightarrow FORK_4)$$
$$FORK_5 = (get^5_5 \rightarrow put^5_5 \rightarrow FORK_5 \mid get^4_5 \rightarrow put^4_5 \rightarrow FORK_5)$$
$$PHIL_1 = (think_1 \rightarrow get^1_2 \rightarrow get^1_1 \rightarrow eat_1 \rightarrow put^1_2 \rightarrow put^1_1 \rightarrow PHIL_1)$$
$$PHIL_2 = (think_2 \rightarrow get^2_2 \rightarrow get^2_3 \rightarrow eat_2 \rightarrow put^2_2 \rightarrow put^2_3 \rightarrow PHIL_2)$$
$$PHIL_3 = (think_3 \rightarrow get^3_4 \rightarrow get^3_3 \rightarrow eat_3 \rightarrow put^3_4 \rightarrow put^3_3 \rightarrow PHIL_3)$$
$$PHIL_4 = (think_4 \rightarrow get^4_4 \rightarrow get^4_5 \rightarrow eat_4 \rightarrow put^4_4 \rightarrow put^4_5 \rightarrow PHIL_4)$$
$$PHIL_5 = (think_5 \rightarrow get^5_1 \rightarrow get^5_5 \rightarrow eat_5 \rightarrow put^5_1 \rightarrow put^5_5 \rightarrow PHIL_5)$$
$$\| DINERS = (FORK_1 \| \ldots \| FORK_5 \| PHIL_1 \| \ldots \| PHIL_5)$$

- Solutions with Elementary Petri Nets.

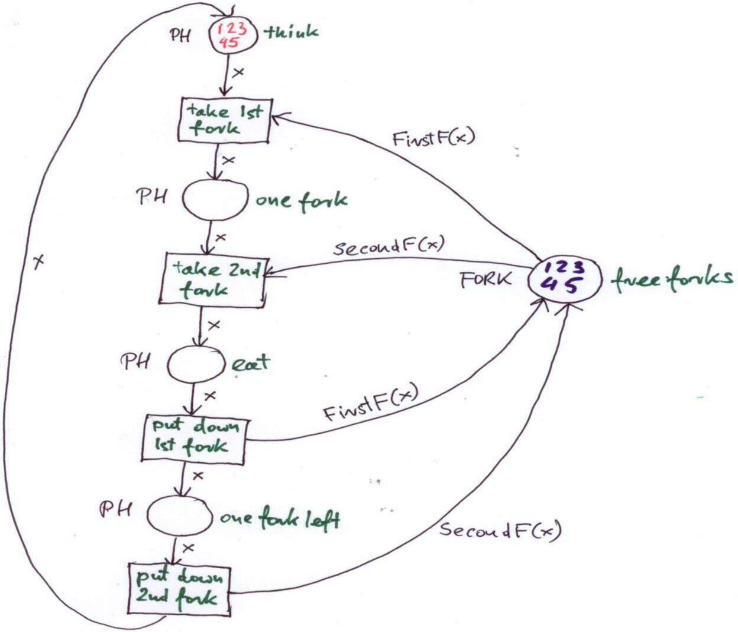
Now we may transform each individual FSP into an appropriate Elementary Petri Net. To simplify net solution (and make it more in 'net spirit'), we may model 'think' and 'eat' by places instead of transitions. For example the nets corresponding to $FORK_1$ and $PHIL_1$ may look as follows:



Now we just need to compose the nets for $FORK_1, \ldots, FORK_5$, $PHIL_1, \ldots, PHIL_5$, by gluing together the same actions. The solution fits one page but barely.

Solution with Coloured Petri nets

colour PH = with ph1 | ph2 | ph3 | ph4 | ph5
colour Fork = with f1 | f2 | f3 | f4 | f5
FirstF : PH $\rightarrow$ FORK, SecondF : PH $\rightarrow$ FORK
FirstFR : PH $\rightarrow$ FORK, SecondFR : PH $\rightarrow$ FORK
var x: PH
'*for philosophers 1, 3 and 5, left fork is first, for philosophers 2 and 4, right fork is first*'
fun FirstF x = case of ph1 $\Rightarrow$ f2 | ph2 $\Rightarrow$ f2 | ph3 $\Rightarrow$ f4 | ph4 $\Rightarrow$ f5 | ph5 $\Rightarrow$ f5
fun SecondF x = case of ph1 $\Rightarrow$ f1 | ph2 $\Rightarrow$ f3 | ph3 $\Rightarrow$ f3 | ph4 $\Rightarrow$ f3 | ph5 $\Rightarrow$ f1
fun FirstFR x = case of ph1 $\Rightarrow$ f2 | ph2 $\Rightarrow$ f2 | ph3 $\Rightarrow$ f4 | ph4 $\Rightarrow$ f5 | ph5 $\Rightarrow$ f5
fun SecondFR x = case of ph1 $\Rightarrow$ f1 | ph2 $\Rightarrow$ f3 | ph3 $\Rightarrow$ f3 | ph4 $\Rightarrow$ f3 | ph5 $\Rightarrow$ f1

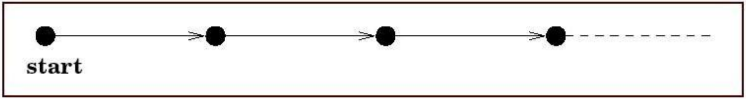

PH
123
45
think
x
take 1st fork
FirstF(x)
x
PH
one fork
x
SecondF(x)
take 2nd fork
FORK
123
45
free forks
x
PH
eat
x
x
FirstF(x)
put down 1st fork
x
PH
one fork left
x
SecondF(x)
put down 2nd fork

Model Checking and Temporal Logic

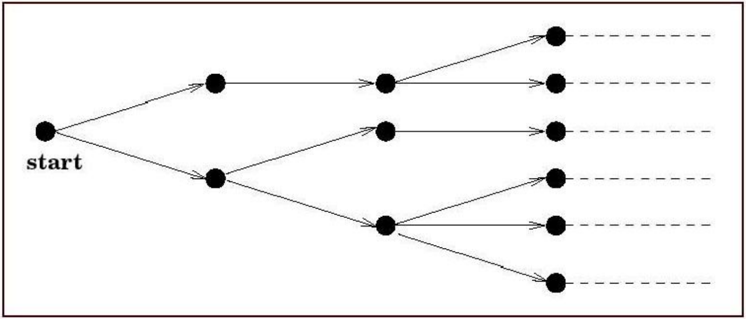
- The model checker outputs the answer "yes" if $\mathcal{M}$ satisfies Φ and "no" otherwise; in the latter case, most model checkers also produce a *trace of system behaviour which causes* this failure.
- There are many *temporal logics*, we concentrate on CTL (Computation Tree Logic) and LTL (Linear Time Logic).
- Time could be *continuous* or *discrete*, we concentrate on *discrete time*.
- $\mathcal{M}$ is not a description of an actual physical system. Models are abstractions that omit lots of real features of a physical systems. *We have similar situation in calculus, mechanics, etc., where we have straight lines, perfect circles, no friction, etc.*

Typical Models of Time

- **Linear Time**: used for Linear Temporal Logic (LTL)



- **Branching time**: used for CTL, CTL* logics, etc.

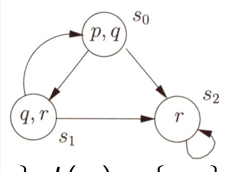


Model

Definition

A **model** $\mathcal{M} = (S, \rightarrow, L)$ for CTL is a set of states S endowed witha transition relation $\rightarrow$ (a binary relation on S), such that every $s \in S$ has some $s' \in S$ with $s \rightarrow s'$ and a labeling function $L : S \rightarrow 2^{Atoms}$.

Example



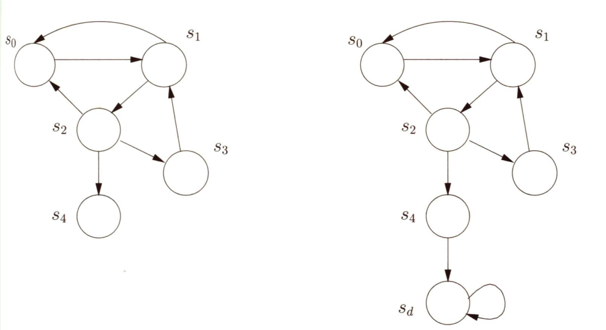
$$L(s_0) = \{p, q\},\ L(s_1) = \{q, r\},\ L(s_2) = \{r\}$$

No deadlock

Definition

“No deadlock” iff for every $s \in S$ there is at least one $s' \in S$ such that $s \to s'$.

Example



A system with a deadlock

A system without a deadlock, s_d is a “deadlock” state

Examples of CTL Formulas

- An upwards traveling elevator at the second floor does not change its direction when it has passengers wishing to go to the fifth floor:

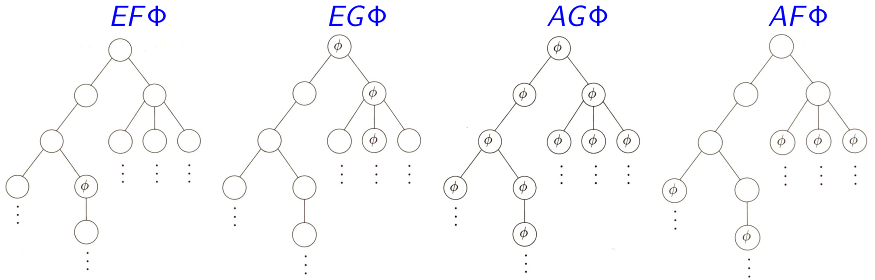
$$AG(floor = 2 \wedge direction = up \wedge ButtonPressed5 \Rightarrow \\ A[direction = up \; U \; floor = 5])$$

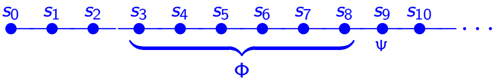
- The elevator can remain idle on the third floor with its doors closed:

$$AG((floor = 3 \wedge idle \wedge door = closed) \Rightarrow \\ EG(floor = 3 \wedge idle \wedge door = closed))$$

- '$floor = 2$', '$direction = up$', $ButtonPressed5$', '$door = closed$', etc. are *names* of *atomic formulas*.

Semantics: Illustrations



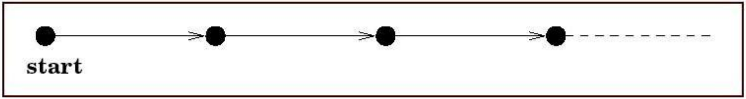


each of the states from s_3 to s_9 satisfies $\Phi\ U\ \Psi$

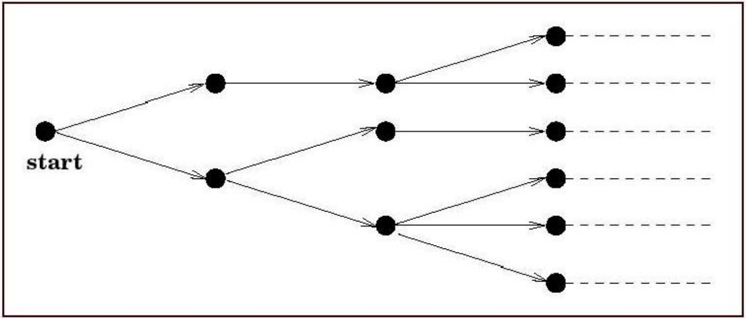
- If the given set of states is finite, then we may compute the set of all states satisfying Φ.
- If $\mathcal{M}$ is obvious, we will write $s \models \Phi$.

Typical Models of Time

- **Linear Time**: used for Linear Temporal Logic (LTL)



- **Branching time**: used for CTL, CTL* logics, etc.



$$\Phi ::= \bot \mid \top \mid p \mid (\neg\Phi) \mid (\Phi \wedge \Phi) \mid (\Phi \vee \Phi) \mid (\Phi \Rightarrow \Phi) \mid$$
$$(G\Phi) \mid (F\Phi) \mid (X\Phi) \mid (\Phi\ U\ \Phi) \mid (\Phi\ W\ \Phi) \mid (\Phi\ R\ \Phi)$$

where p ranges over atomic formulas/descriptions.

- $\bot$ - false, $\top$ - true
- $G\Phi, F\Phi, X\Phi, \Phi\ U\ \Phi, \Phi\ W\ \Phi, \Phi\ R\ \Phi$ are **temporal connections**.
- X means "neXt moment in time"
- F means "some Future moments"
- G means "all future moments (Globally)"
- U means "Until"
- W means "Weak-until"
- R means "Release"
- An LTL formula is evaluated on a path, or a set of paths.
- A set of paths satisfies Φ if every path in the set satisfies Φ.
- Consider the path $\pi \stackrel{\text{df}}{=} s_1 \rightarrow s_2 \rightarrow \ldots$.
  We write π^i for the suffix starting at s_i, i.e. π^i is
  $s_i \rightarrow s_{i+1} \rightarrow s_{i+2} \rightarrow \ldots$.

Practical Patterns of LTL Specifications (1)

What kind of practically relevant properties can we check with formulas of LTL?
Suppose atomic descriptions include some words as busy, requested, ready, etc.

- It is impossible to get a state where started holds but ready does not hold:

  $G\neg(started \wedge \neg ready)$

- For any state, if a request (of some resource) occurs, then it will eventually be acknowledged:

  $G(requested \Rightarrow F\ acknowledged)$

- A certain process is enabled infinitely often on every computation path:

  $GF\ enabled$

- On all path, a certain process will eventually be permanently deadlocked:

  $FG\ deadlock$

Impossible LTL Specifications

There are some things which are **not** possible to say in LTL, however. One big class of such things are statements which assert the existence of a path, such as these ones:

- For any state it is *possible* to get a **restart** state (i.e., there is a path from all states to a state satisfying **restart**).
- The lift *can* remain idle on the third floor with its doors closed (i.e., from the state in which it is on the third floor, there is a path along it stays there).

LTL cannot express these because it cannot directly assert the existence of path. CTL has operators for quantifying over paths, and **can** express these properties.

CTL vs LTL

- It is possible to get a state where started holds but ready does not hold:
  CTL: $EF(started \wedge \neg ready)$
  LTL: $G\neg(started \wedge \neg ready)$
- For any state, if a request (of some resource) occurs, then it will eventually be acknowledged:
  CTL: $AG(requested \Rightarrow AF\ acknowledged)$
  LTL: $G(requested \Rightarrow F\ acknowledged)$
- A certain process is enabled infinitely often on every computation path:
  CTL: $AG(AF\ enabled)$
  LTL: $GF\ enabled$
- Whatever happens, a certain process will eventually be permanently deadlocked:
  CTL: $AF(AG\ deadlock)$
  LTL: $FG\ deadlock$

CTL* Logic

It allows nested modalities and boolean connectives before applying the path quantifiers E and A.

- $A[(p\ U\ r) \vee (q\ U\ r)]$: along all paths, either p is true until r, or q is true until r.
  $\not\equiv A[(p \vee q)\ U\ r]$
  It can be expressed in CTL, but it is not easy.
- $A[X\ p \vee X\ X\ p]$: along all paths, p is true in the next state, or the next but one.
  $\not\equiv AX\ p \vee AXAX\ p$
  It **cannot** be expressed in CTL.
- $E[GF\ p]$: there is a path along which p is infinitely often true.
  $\not\equiv EGEF\ p$
  It **cannot** be expressed in CTL.

CTL* Syntax

The syntax of CTL* involves two classes of formulas:

- **state formulas**, which are evaluated in states:

  $$\Phi ::= \bot \mid \top \mid p \mid (\neg\Phi) \mid (\Phi \wedge \Phi) \mid (\Phi \vee \Phi) \mid (\Phi \Rightarrow \Phi) \mid A[\alpha] \mid E[\alpha]$$

  where p is any atomic formula and α is any path formula.

- **path formulas**, which are evaluated along paths:

  $$\alpha ::= \Phi \mid (\neg\alpha) \mid (\alpha \wedge \alpha) \mid (\alpha \vee \alpha) \mid (\alpha \Rightarrow \alpha) \mid (\alpha\ U\ \alpha) \mid (G\ \alpha) \mid (F\ \alpha) \mid (X\ \alpha)$$

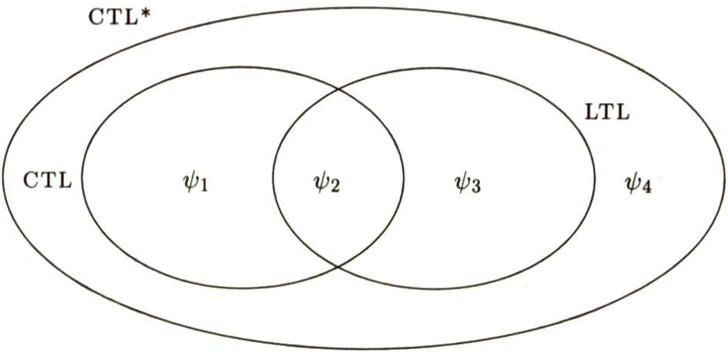
  where Φ is any state formula.

LTL, CTL vs CTL*

- LTL is a subset of CTL*.
  Although the syntax of LTL does not include A, E, the semantic viewpoint of LTL is that we consider all path. Therefore, the LTL formula α is equivalent to the CTL* formula $A[\alpha]$.
- CTL is a subset of CTL*.
  CTL is a fragment of CTL* in which we restrict the form of path formulas to:

$$\alpha ::= (\Phi\ U\ \Phi) \mid (G\ \Phi) \mid (F\ \Phi) \mid (X\ \Phi)$$

LTL, CTL vs CTL*



Problem: Express in LTL and CTL: 'Whenever p is followed by q (after some finite amount of steps), then the system enters an 'interval' in which no r occurs until t'.

Solution:

- The process of translating informal requirements into formal specifications is subject to various pitfalls.
- One of them is simply ambiguity.
- For example it is unclear whether "after some finite steps" means "at least one, but finitely many", or whether zero steps are allowed as well.
- It may also be debatable what "then" exactly means in "... then the system enters...".
- We chose to solve this problem for the case when zero steps are not admissible, mostly since "followed b" suggest a real state transition to tale place.

Problem: Express in LTL and CTL: 'Whenever p is followed by q (after some finite amount of steps), then the system enters an 'interval' in which no r occurs until t'.

Solution (continued):
The LTL formula is the following

$$G(p \implies XG(\neg q \vee \neg r U t)),$$

while an equivalent CTL formula is:

$$AG(p \implies AXAG(\neg q \vee A\neg r U t])).$$

It says: At any state, if p is true, then at any state which one can reach with at least one state transition from here, either q is false, or r is false until t becomes true (for all continuations of the computation path).
This is evidently the property we intended to model.
Various other "equivalent" solutions can be given.

Problem: Express in LTL and CTL: 'Between the events q and r, p is never true'

Solution:

- Ambiguities: Is the case when r or q never happens allowed?
- We assume that it is not.
- What exactly "between" means?
- We assume "between" is "closed interval" so p is false in the state that holds q and in the state that holds r.

LTL: $G(Fq \wedge Fr \wedge (\neg q \vee (\neg p U r))$

CTL: $AG(AFq \wedge AFr) \wedge AG(q \implies A(\neg p U r))$

Problem: Consider the CTL formula

$$AG(p \implies AF(s \wedge AX(AF\ t))).$$

Explain what exactly it expresses in terms of the order of occurrences of events p, s and t.

Solution:

- For every history, if p occurs then p may occur simultaneously with s (as "future includes present") or s occurs after p, and t always occurs after p.
- If p does not occur in a history, then any order between s and t is allowed.

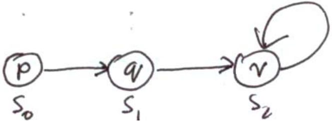
Problem: Which of the following pairs of CTL formulas are equivalent? Prove either case.

(a) $EF\ \Phi$ and $EG\ \Phi$
(b) $EF\ \Phi \vee EF\ \Psi$ and $EF(\Phi \vee \Psi)$
(c) $AF\ \Phi \vee AF\ \Psi$ and $AF(\Phi \vee \Psi)$

- For non-equivalence, we need to show a counterexample.

Problem (a): $EF\ \Phi\ \not\equiv EG\ \Phi$.

Consider the below model:



We have $s_0 \models EF\ r$ since $L(s_1) = \{r\}$, but $s_0 \not\models EG\ r$ since $r \notin L(s_0) \wedge r \notin L(s_1)$.

Problem (b): We have $s \models EF\ \Phi \vee EF\ \Psi$ iff $s \models EF(\Phi \vee \Psi)$.

Proof:

($\Rightarrow$) First assume that $s \models EF\ \Phi \vee EF\ \Psi$.

Then, without loss of generality, we may assume that $s \models EF\ \Phi$ (the other case is shown in the same manner).

This means that there is a future state s_n, reachable from s, such that $s_n \models \Phi$. But then $s_n \models \Phi \vee \Psi$ follows.

But this means that there is a state reachable form s which satisfies $\Phi \vee \Psi$.

Thus $s \models EF(\Phi \vee \Psi)$.

($\Leftarrow$) Assume $s \models EF(\Phi \vee \Psi)$.

Then there exists a state s_m, reachable from s, such that $s_m \models \Phi \vee Psi$.

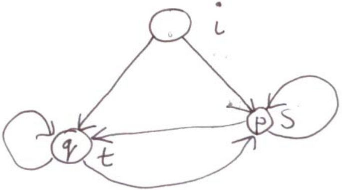
Without loss of generality, we may assume that $s_m \models \Phi$.

But then we can conclude that $s \models EF\Phi$, as s_m is reachable from s.

Therefore, we also have $s \models EF\ \Phi \vee EF\ \Psi$.

Problem (c): While we have that $s \models (AF\ \Phi \vee AF\ \Psi$ implies $s \models AF(\Phi \vee \Psi)$ the converse is not true.

Proof:
Consider the model:



Clearly $i \models AF(p \vee q)$ since $L(s) \cap \{p, q\} \neq \emptyset$ and $L(t) \cap \{p, q\} \neq \emptyset$.
But $i \not\models AF\ p$, see the path $i \rightarrow t \rightarrow t \rightarrow t \rightarrow \ldots$, and $i \not\models AF\ q$, see the path $i \rightarrow s \rightarrow s \rightarrow s \rightarrow \ldots$.

Problem: Use the definition of $\models$ to explain why $s \models AG\ AF\ \Phi$ means "Φ is true infinitely often along every paths starting at s".

Solution:

- $s \models AG\ AF\ \Phi$ means that for every s' reachable from s, i.e. $s \to^* s'$, we have $s' \models AF\ \Phi$.
- $s' \models AF\ \Phi$ means that for each path starting from s', there exists s'' reachable from s', i.e. $s' \to^* s''$ such that $s'' \models \Phi$.
- For a given path π_s starting from s, let $next_\Phi^{\pi_s}(s)$ be the state s_Φ such that $s_\Phi \models \Phi$ and n such that $s \to^n s_\Phi$ is minimal, i.e. the distance between s and $next_\Phi^{\pi_s}(s)$ is minimal.
- In general $next_\Phi^{\pi_s}(s)$ may not exists, but is that case for each s' such that $s \to^* s'$, for $next_\Phi^{\pi_{s'}}(s')$ always does exist.
- Define $s^1 = next_\Phi^{\pi_s}(s), s^2 = next_\Phi^{\pi_{s_1}}(s_1), s^3 = next_\Phi^{\pi_{s_2}}(s_2), \ldots$
- Such a sequence always exists no matter which path π_{s_i} we chose and this sequence is always infinite.

Problem: The meaning of temporal operators AU, EU, AG, EG, AF, and EF was defined to be such that "the future includes the present". For example, $EF\ p$ is true for a state if p is true for that state already. Often one would like corresponding operators such the "the future excludes the present". For instance $newAG\ \Phi$ could be defined as $AX(AG\ \Phi)$.
Define $newEG$, $newAF$, $newAU$, $newEU$.
Solutions:

$newEG\ \Phi$: $EX(AG\ \Phi)$
$newAF\ \Phi$: $AX(AF\ \Phi)$

- As for the U connective, we basically want to maintain the nature of the $\Phi_1\ U\ \Phi_2$ pattern, but what changes is that we ban the extreme case of having Φ_2 at the first state.
- Thus we have to make sure that Φ_1 is true in the current state and conjoin this with the shifted AU, respectively EU operators.

$newAU$: $\Phi_1 \wedge AX(A[\Phi_1\ U\ \Phi_2])$
$newEU$: $\Phi_1 \wedge EX(E[\Phi_1\ U\ \Phi_2])$

Problem: Sometimes, in formal logic, we are forced to prove things that intuitively look obvious.

Show that a CTL formula Φ is true on infinitely many states of a computation path $s_0 \to s_1 \to s_2 \to \dots$ iff for all $n \geq 0$ there is some $m \geq n$ such that $s_m \models \Phi$.

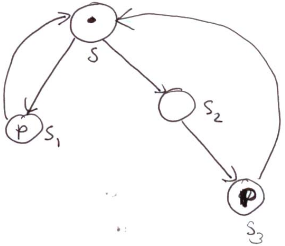
Solution:

$(\Rightarrow)$ Let Φ is true on infinitely many states of a computation path $s_0 \to s_1 \to s_2 \to \dots$. Suppose that the negation of our claim is true. This means that there exists some $n \geq 0$ such that for all $m > n$ we have $s \not\models \Phi$. But then Φ could only be true on finitely many states of that path, namely at most at the states $s_0, s_1, s_2, \dots s_{n-1}$.

$(\Leftarrow)$ Suppose that for every $n \geq 0$, there is some $m > n$ with $s_m \models \Phi$. Assume that Φ is only true at finitely many states of that path. The there has to be a maximal number n_0 such that no s_m with $m > n_0$ satisfies Φ. But this is a contradiction to the assumption "for all $n \geq 0$", so in particular n_0.

Problem: Show that the following CTL* formulas are **not** equivalent: $A[X\ p \vee XX\ p]$ and $AX\ p \vee AX\ AX\ p$.

Solution.
One can show that $A[X\ p \vee XX\ p]$ implies $AX\ p \vee AX\ AX\ p$ (standard proof, I will omit).
Consider the model:



$s \models A[X\ p \vee XX\ p]$ since every path has to turn left, i.e. $X\ p$, or right. i.e. $XX\ p$.
$s \not\models AX\ p$ (right turn), see the path $s \to s_2 \to s_3 \to \ldots$
$s \not\models AXAX\ p$ (left turn), see the path $s \to s_1 \to s \to s_2 \to \ldots$

Problem: Invent CTL formulas equivalent for the following CTL* formulas

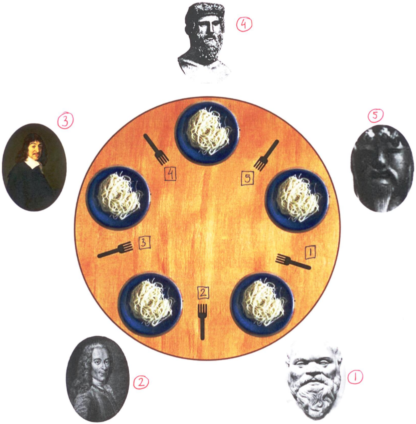
(a) $E[F\ p \wedge (q\ U\ r)]$
(b) $A[F\ p \implies F\ q]$

Solutions:

(a) $E[F\ p \wedge (q\ U\ r)]$ is equivalent to
$E[q\ U\ (p \wedge E[q\ U\ r])] \vee [q\ U\ (r \wedge EF\ p)]$.
(b) $A[F\ p \implies F\ q]$ is equivalent to $\neg E[F\ p \wedge G \neg q]$,
which we can write as $\neg E[\neg q\ U\ (p \wedge EG\ \neg q]$, which is in CTL.

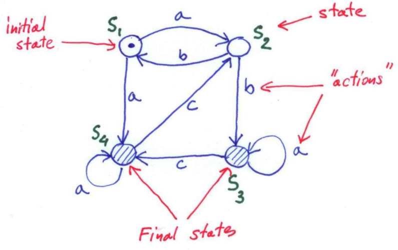
Dining Philosophers

- Five philosophers sit around a circular table. Each philosopher spends his life alternately thinking and eating. To eat, a philosopher needs two forks, but unfortunately there are only five forks on the circulr table and each philosopher is only allowed to use the two forks nearest to him.

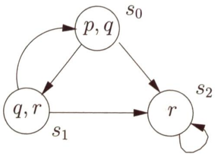


Finite Automata vs Kripke Structures

- For Finite Automata, transitions are *actions*, while states do not have standard interpretations.

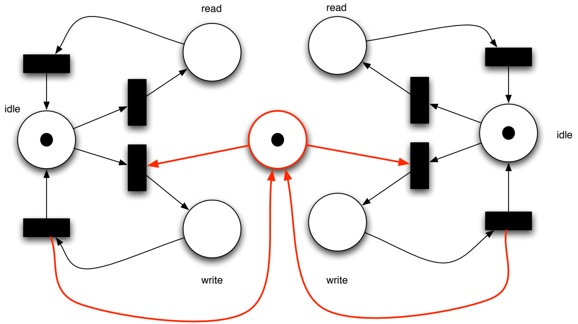


- For Kripke Structures, transitions are just changes of states:



- Orthogonal interpretations!

Example I



Add a lock to ensure mutual exclusion

- Reachability graphs for Petri nets with actions in places correspond intuitively to Kripke structures!

Kripke Structure for Dining Philosophers

- Atomic Predicates(for various cases):
  Ph_iT - philosopher i is thinking
  Ph_iF_j - philosopher i has fork j
  $Ph_iF_{\{j,k\}}$ - philosopher i has forks j and k
  Ph_iE - philosopher i is eating
  Ph_iTD - philosopher i has a ticket to dining room
  Ph_iD - philosopher i is in dining room
  F_i - fork i is free
  Tic_i - i tickets remain
  tic_i - ticket number i not taken
  ...
- Some properties of atomic predicates.
  $Ph_iF_{\{j,k\}} = Ph_iF_{\{k,j\}}$ for all j, k, and always $j \neq k$.
  For standard version Ph_iF_j implies $i = j \vee j = i + 1 \mod 5$.
  For standard version $Ph_iF_{\{j,k\}}$ implies $i = j \wedge k = i + 1 \mod 5$.
  For each state s, if $Ph_iF_j \in L(s)$ and $F_k \in L(s)$ then $j \neq k$.
  For each state s, if $Ph_iF_j \in L(s)$ and $Ph_kF_l \in L(s)$ then $i \neq k \wedge j \neq l$.
  ...

- States(for Model Checking the states are global)
  If s_0 is the initial state than
  $L(s_0) = \{Ph_1T, Ph_2T, Ph_3T, Ph_4T, Ph_5T, F_1, F_2, F_3, F_4, F_5\}$.
  For 'deadlock' state s_d,
  $L(s_d) = \{Ph_1F_1, Ph_2F_2, Ph_3F_3, Ph_4F_4, Ph_5F_5\}$
  A legal state s with 2 philosophers eating and 3 philosophers thinking, $L(s) = \{Ph_1E, Ph3E, Ph_2T, Ph_4T, Ph_5T, F5\}$
  ...
- States are standardly defined by their labels, i.e. atomic predicates that hold in them.
- Extra atomic control predicates may be added, as *turn0, turn1* for mutual exclusion solution.
- States that are 'impossible' as s with $L(s) = \{Ph_1E, Ph3E, Ph_2T, Ph_4T, Ph_5T, F4, F5\}$ are allowed, and often useful!

- Transitions

  For example $s_0 \to s_1$, when
  $L(s_0) = \{Ph_1T, Ph_2T, Ph_3T, Ph_4T, Ph_5T, F_1, F_2, F_3, F_4, F_5\}$
  and
  $L(s_1) = \{Ph_1F1, Ph_2T, Ph_3T, Ph_4T, Ph_5T, F_2, F_3, F_4, F_5\}$

  For example $s_0 \to s_1$, when s_0 as above and
  $L(s_1) = \{Ph_1F_{12}, Ph_2T, Ph_3T, Ph_4T, Ph_5T, F_3, F_4, F_5\}$, for a version with simultaneous pick up of two forks.

- We need some simple programming language to represent states and transitions, so we can program an appropriate Kripke structure.