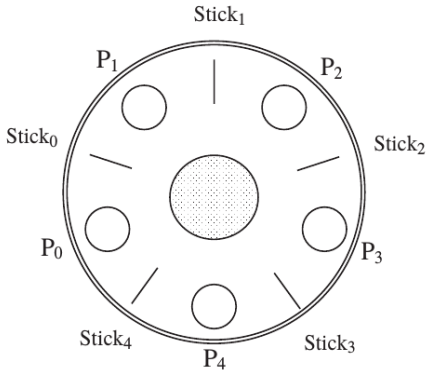
Linear-Time properties

PROPERTIES OF TS

Dining philosophers again!

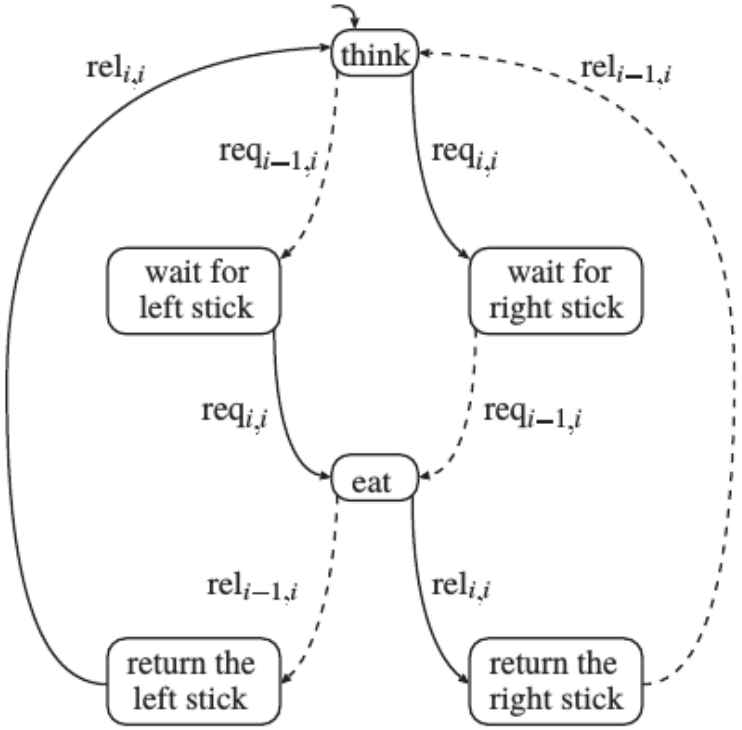


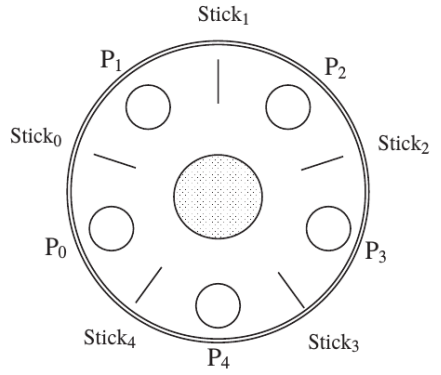
To take some rice out of the bowl, a philosopher needs two chopsticks.
In between two neighboring philosophers, however, there is only a single chopstick.
Thus, at any time only one of two neighboring philosophers can eat.

A deadlock scenario occurs when all philosophers possess a single chopstick.

Problem: design (and verify) a protocol such that the complete system is *deadlock-free*, i.e., at least one philosopher can eat and think infinitely often.
Additionally, a fair solution may be required with each philosopher being able to think and eat *infinitely often* (freedom of individual starvation).

A Philosopher



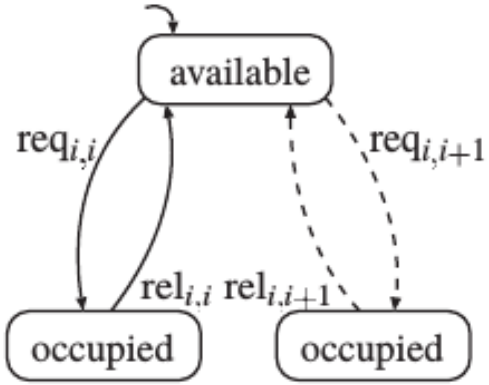


Process Phil$_i$:
Philosopher will nondeterm. try to grab one of the two nearby sticks, with no preferred order

For clarity, solid arrows depict the synchronizations with the i-th stick, dashed arrows refer to comm. with the (i−1)-th stick.

req$_{j,i}$ is Phil$_i$ asking for Stick$_j$
(i.e, req$_{i-1,i}$ is Phil$_i$ asking for Stick$_{i-1}$ and req$_{i,i}$ is Phil$_i$ asking for Stick$_i$)
NB: All operations i-1 are modulo 5.

A wrong solution for the Sticks



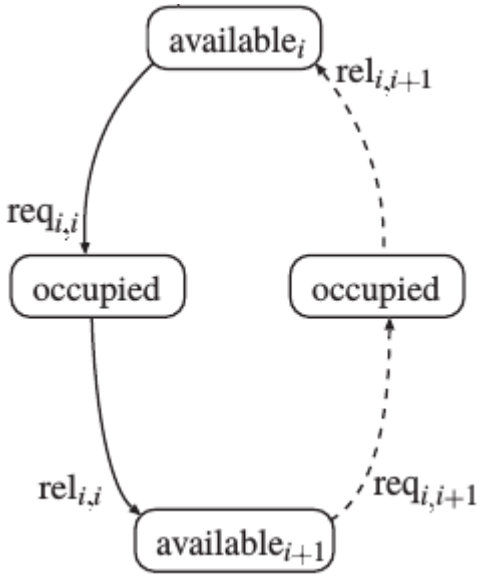
Process $Stick_i$ with which the philosophers synchronize via actions *req* and *rel*.
It ensures mutual exclusion: it prevents $Phil_i$ from picking up the i-th stick when $Phil_{i+1}$ is using it and vice versa.

When a Stick is requested (e.g, $req_{i,i+1}$: $Phil_{i+1}$ asking for Stick i), if it is available, becomes occupied

The complete system is of the form:

$$Phil_4 \parallel Stick_3 \parallel Phil_3 \parallel Stick_2 \parallel Phil_2 \parallel Stick_1 \parallel Phil_1 \parallel Stick_0 \parallel Phil_0 \parallel Stick_4$$

A correct solution for Stick $_i$



A possible solution: make the sticks available for only one philosopher at a time.
In state *available* $_i$ only Phil $_i$ is allowed to pick up the i-th stick.
In state *available* $_{i+1}$ only Phil $_{i+1}$ is allowed to pick up the i-th stick.

Deadlock is avoided if some sticks (e.g., the first, the third, and the fifth stick) start in state *available* $_i$, while the remaining sticks start in state *available* $_{i+1}$

It can be verified that this solution is deadlock- and starvation-free.

What are deadlock and starvation?

- What kind of properties are deadlock-freedom and starvation freedom?
- How can we verify them?

Linear-Time Properties

- A linear time property specifies a desired behavior of a system.
  - Remind: A run (or execution) of a TS is an alternating sequence of states and actions
  - We often are not interested in actions (used mainly for internal communication)
- Need to return on concept of **trace**
  - we only want to see the sequence of atomic propositions
  - Traces can also be infinite (non-terminating behavior)
  - Remind: Given (infinite) run $r_{\sigma} = s_0\ i_1\ s_1\ i_2\ s_2\ \ldots$, its *trace* is the sequence of AP subsets $L(s_0)\ L(s_1)\ L(s_2)\ \ldots$
- Given a TS without terminal states:

  Trace(TS) is the set of all its infinite traces.

  Trace(s) is the set of infinite traces from state s.

Infinite words and traces

- The set of *infinite* words over alphabet A is denoted as A^{ω}
- Analogy with the set of finite words over alphabet A, denoted as A^{*}
- A Linear-Time (LT) property over a set AP of atomic propositions is a subset of $(2^{AP})^{\omega}$
- An infinite word: {red1,green2} {red1, red2} {green1} ∅ ∅ {green2} …
- A LT property is not a formula but a **set** of infinite words over the «alphabet» 2^{AP}

$$TS = (S, Act, \rightarrow, I, AP, L) \ satisfies \ P, \text{ denoted } TS \models P$$
$$\text{iff } Traces(TS) \subseteq P.$$
$$\text{State } s \in S \text{ satisfies } P, \text{ notation } s \models P, \text{ whenever } Traces(s) \subseteq P.$$

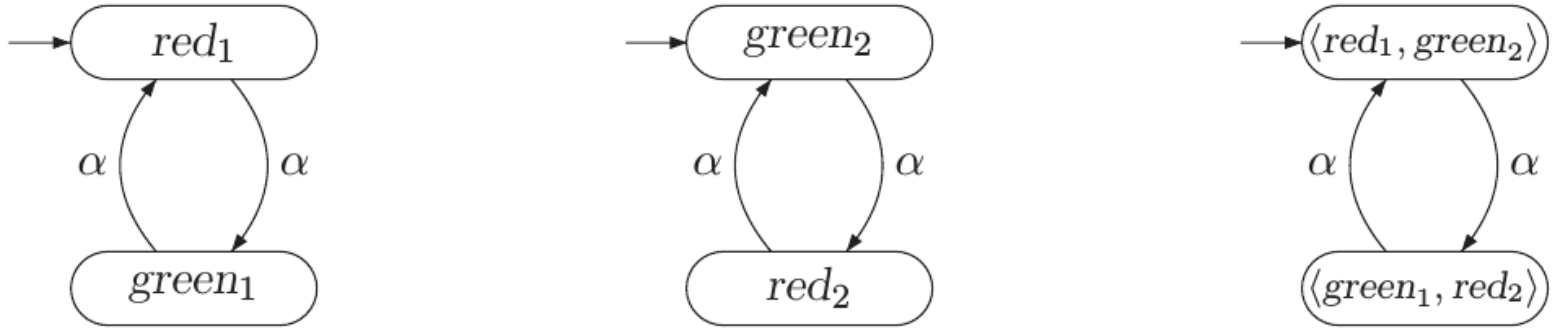


Figure 3.7: Two fully synchronized traffic lights (left and middle) and their parallel composition (right).

"The first traffic light is infinitely often green".

This LT property corresponds to the set of infinite words of the form $A_0\,A_1\,A_2\,\ldots$ over 2^{AP}, such that $green_1 \in A_i$ holds for infinitely many i. For example, P contains the infinite words

$$\{\, red_1, green_2 \,\}\,\{\, green_1, red_2 \,\}\,\{\, red_1, green_2 \,\}\,\{\, green_1, red_2 \,\}\ldots,$$

$$\varnothing\,\{\, green_1 \,\}\,\varnothing\,\{\, green_1 \,\}\,\varnothing\,\{\, green_1 \,\}\,\varnothing\,\{\, green_1 \,\}\,\varnothing\ldots$$

$$\{\, red_1, green_1 \,\}\,\{\, red_1, green_1 \,\}\,\{\, red_1, green_1 \,\}\,\{\, red_1, green_1 \,\}\ldots \quad \text{and}$$

$$\{\, green_1, green_2 \,\}\,\{\, green_1, green_2 \,\}\,\{\, green_1, green_2 \,\}\,\{\, green_1, green_2 \,\}\ldots$$

The infinite word $\{\, red_1, green_1 \,\}\,\{\, red_1, green_1 \,\}\,\varnothing\,\varnothing\,\varnothing\,\varnothing\ldots$ is not in P as it contains only finitely many occurrences of $green_1$.

Another example of property

"The traffic lights are never both green simultaneously".

This property is formalized by the set of infinite words of the form $A_0\, A_1\, A_2 \ldots$ such that either $green_1 \notin A_i$ or $green_2 \notin A_i$, for all $i \geqslant 0$. For example, the following infinite words are in P':

$$\{\, red_1, green_2 \,\}\{\, green_1, red_2 \,\}\{\, red_1, green_2 \,\}\{\, green_1, red_2 \,\}\ldots,$$

$$\varnothing\,\{\, green_1 \,\}\,\varnothing\,\{\, green_1 \,\}\,\varnothing\,\{\, green_1 \,\}\,\varnothing\,\{\, green_1 \,\}\,\varnothing\ldots \qquad \text{and}$$

$$\{\, red_1, green_1 \,\}\{\, red_1, green_1 \,\}\{\, red_1, green_1 \,\}\{\, red_1, green_1 \,\}\ldots,$$

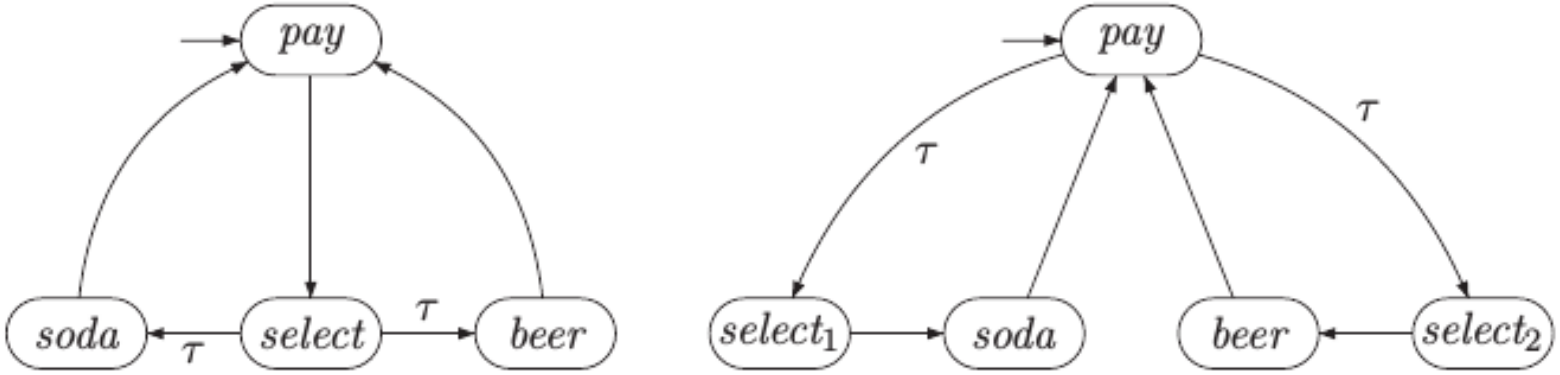
whereas the infinite word $\{\, red_1\, green_2 \,\}\{\, green_1, green_2 \,\}, \ldots$ is not in P'.

Trace equivalence
(aka language equivaence)

Definition 3.17. Trace Equivalence

Transition systems TS and TS' are *trace-equivalent* with respect to the set of propositions AP if $Traces_{AP}(TS) = Traces_{AP}(TS')$. [4]

For simplicity, the observable action labels of transitions have been omitted.



Although the two vending machines behave differently, they exhibit the same traces *when considering AP*= {pay , soda, beer}: (NB: we removed {…} for singletons)
for both machines traces are alternating sequences of *pay* and either *soda* or *beer*.

The vending machines are thus trace-equivalent.

Trace equivalence and linear time properties

Corollary 3.18. Trace Equivalence and LT Properties

Let TS and TS' be transition systems without terminal states and with the same set of atomic propositions. Then:

$$Traces(TS) = Traces(TS') \iff \text{TS and } TS' \text{ satisfy the same LT properties.}$$

There thus does not exist an LT property that can distinguish between trace-equivalent transition systems. Stated differently, in order to establish that the transition systems TS and TS' are *not* trace-equivalent it suffices to find one LT property that holds for one but not for the other.

Invariants as LT properties

- A LT property P over AP is an **invariant** if there is a propositional logic formula Φ over AP such that P is the set of traces of the form $A_0\ A_1\ A_2,\ldots$, such that A_j satisfies Φ for every $j>=0$
- Ex. mutual exclusion is invariant

- Checking an invariant property is «easy»:
  - Checking an invariant for the propositional formula Φ amounts to checking the validity of Φ in every state that is reachable from some initial state
  - A slight modification of standard graph traversal algorithms like depth-first search (DFS) or breadth-first search (BFS) will do, provided the given transition system TS is finite.
  - Complexity is linear in the number of states

Safety properties

- A safety property P is an LT property over AP such that *any infinite word σ where P does **not** hold contains a bad prefix.*
  - A bad prefix is a finite prefix σ' where the “bad thing” has already happened, and thus no infinite word that starts with this prefix σ' fulfills P .

An LT property P_{safe} over AP is called a *safety* property if for all words $\sigma \in (2^{AP})^{\omega} \setminus P_{safe}$ there exists a finite prefix $\widehat{\sigma}$ of σ such that

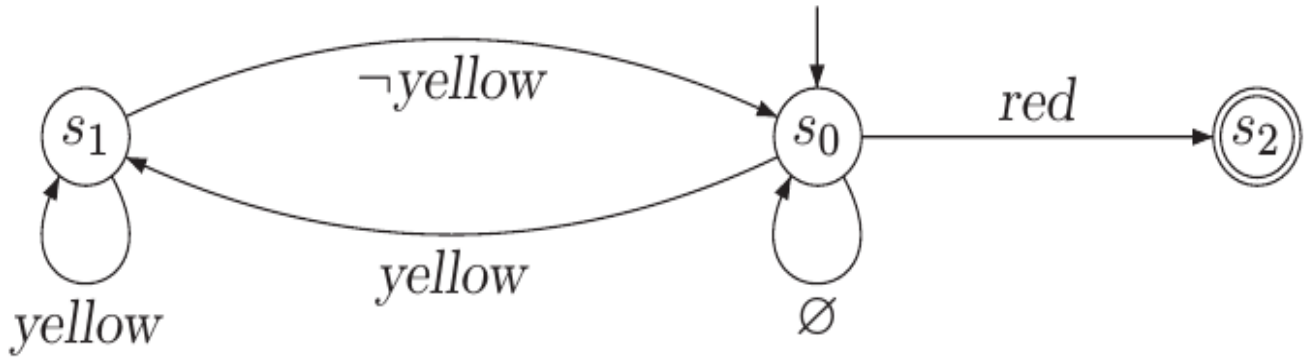
$$P_{safe} \cap \left\{ \sigma' \in (2^{AP})^{\omega} \mid \widehat{\sigma} \text{ is a finite prefix of } \sigma' \right\} = \varnothing.$$

Examples

- All invariants are safety properties
- A non-invariant safety property:
  - We consider a specification of a traffic light with the usual three phases “red”, “green”,and “yellow”.
  - "each red phase should be immediately preceded by a yellow phase”: it is a safety property but not an invariant.
  - A finite trace violating the property: {yellow } { yellow } { red } { red }
  - A finite trace satisfying the property: { yellow } { red }

Expressing LT properties on TS

- Properties to be checked for transition systems can be expressed in many ways
- The most used approach is to focus on *regular properties*.
- For instance, the "bad property" that red is not always preceded by yellow can be defined by a *finite automaton on finite words* (over state labels)



Liveness properties

- A liveness property (over AP) is defined as **an LT property that does not rule out any prefix.**
  - Thus, the set of finite traces of a system are of no use at all to decide whether a liveness property holds or not.
  - Intuitively speaking, it means that any finite prefix can be extended such that the resulting infinite trace satisfies the liveness property
    - For instance, starvation-freedom cannot be ruled out by looking a finite prefixes: maybe that "starving" process will enter the critical section later
  - Instead for safety properties it suffices to have one finite trace (the "bad prefix") to conclude that a safety property is refuted.
- A liveness property can thus be violated *only in an infinite trace*

Example: a mutual exclusion algorithm

- Mutual exclusion is safety property (even an invariant)
- However, mutex can be achieved just having one process entering its critical section and never exiting it!
  - System is not in deadlock, since that process will not be blocked
- Typical liveness properties:
  - (eventually) each process will eventually enter its critical section;
  - (repeated eventually) each process will enter its critical section infinitely often;
  - (starvation freedom) each waiting process will eventually enter its critical section.

Formal definition

LT property P_{live} over AP is a *liveness* property whenever $pref(P_{live}) = (2^{AP})^*$.

Thus, a liveness property (over AP) is an LT property P such that each finite word can be extended to an infinite word that satisfies P. Stated differently, P is a liveness property if and only if for all finite words $w \in (2^{AP})^*$ there exists an infinite word $\sigma \in (2^{AP})^\omega$ satisfying $w\sigma \in P$.

Safety vs. Liveness

- Are safety and liveness properties disjoint? **YES**

The only LT property over AP that is both a safety and a liveness property is $(2^{AP})^{\omega}$.

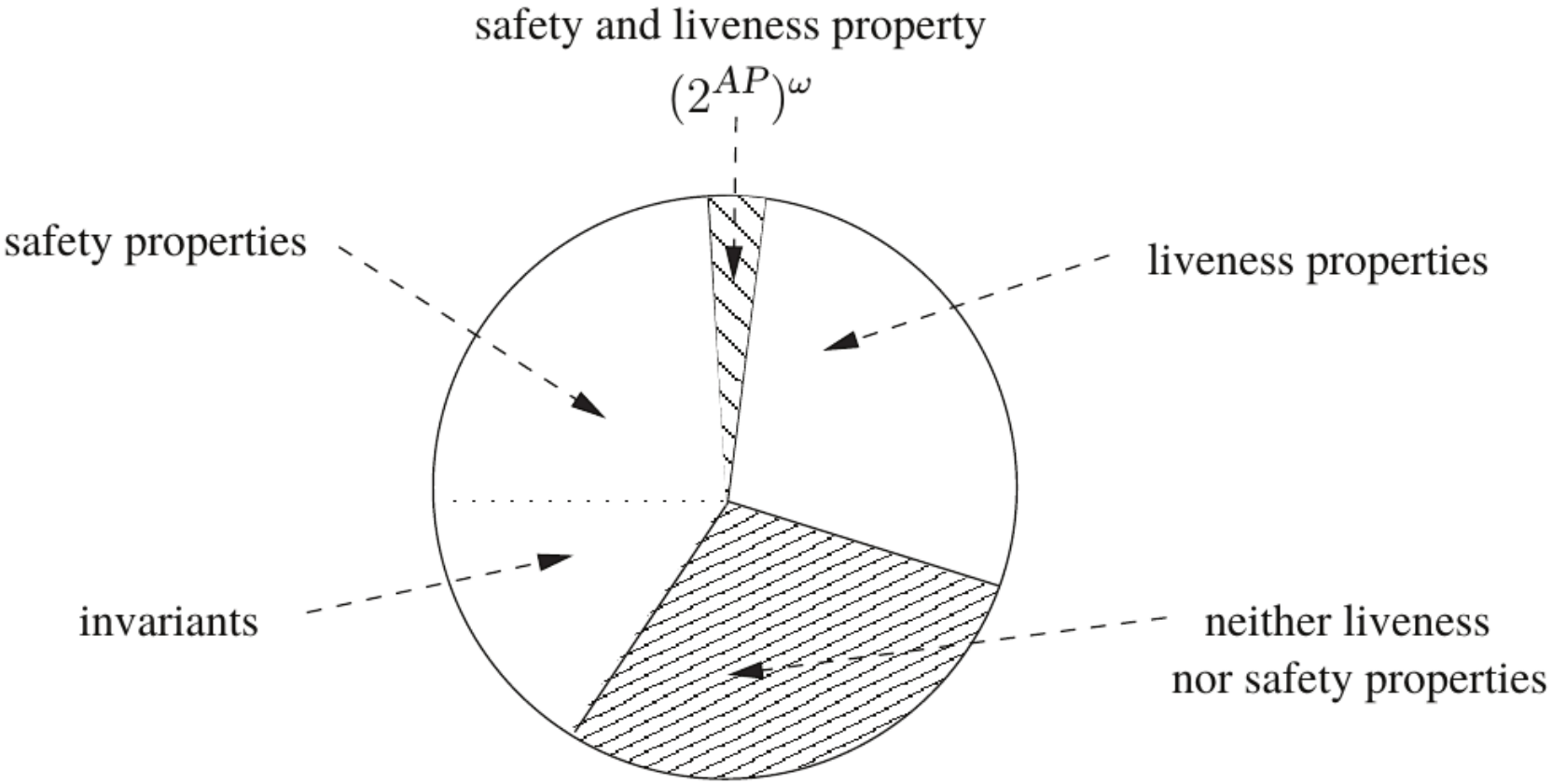
- Is any linear-time property either a safety or a liveness property? **NO**
  - "the machine provides beer infinitely often after initially providing soda three times in a row"
- **but…**

Theorem 3.37. Decomposition Theorem

For any LT property P *over* AP *there exists a safety property* P_{safe} *and a liveness property* P_{live} *(both over* AP*) such that*

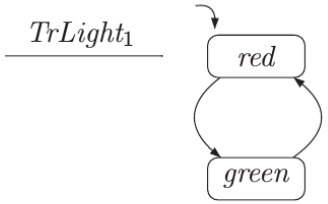
$$P = P_{safe} \cap P_{live}.$$

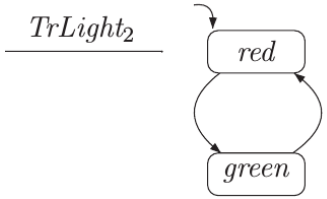
Safety vs. Liveness

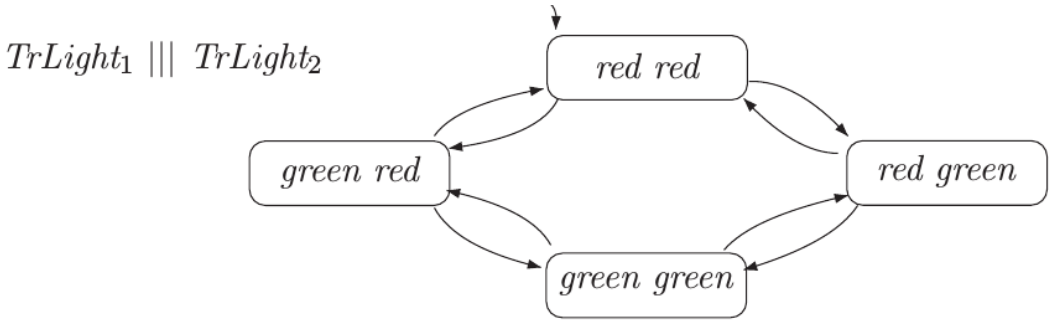


FAIRNESS

Interleaving of 2 independent Traffic lights







"Both traffic lights are infinitely often green" is not satisfied, since
{ red1, red2} { green1, red2} { red1, red2} { green1, red2} . .
is a trace of TS where only the first traffic light is infinitely often green.

Fairness constraints

- The information whether each traffic light switches color infinitely often is lost by means of interleaving.
  - The trace in which only the first traffic light is acting while the second light seems to be completely stopped is formally a trace of the transition system TrLight1 ||| TrLight2.
  - However, **it does not represent a realistic behavior** as in practice no traffic light is infinitely faster than another
  - (there is no “time” over transitions).
- We need to add so-called *fairness constraints* to rule out those unrealistic behaviors

Different Fairness Constraints

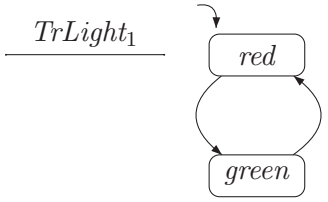
- **Unconditional fairness**: e.g.,“Every process gets its turn infinitely often.”
- **Strong fairness**: e.g., “Every process that is enabled infinitely often gets its turn infinitely often.”
- **Weak fairness**: e.g., “Every process that is continuously enabled from a certain time instant on gets its turn infinitely often.”
  - “is enabled” means “ready to execute (a transition)”.
  - “gets its turn” stands for the execution of an arbitrary transition.
  - Unconditional fairness ⇒ strong fairness ⇒ weak fairness

Intuition

- Unconditional fairness («impartiality»): ex. a process enters its critical section infinitely often (no further conditions)
- Strong fairness («compassion»): if an activity is infinitely often enabled—but there may be finite periods during which the activity is not enabled—then it will be executed infinitely often.
- Weak fairness («justice») means that if an activity, e.g., a transition in a process or an entire process itself, is continuously enabled—no periods are allowed in which the activity is not enabled—then it has to be executed infinitely often.

Fairness constraints

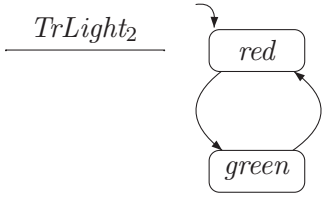
- A fairness constraint is **a set of set of actions** that must take place with a fairness assumption (strong, weak, unconditional)



Consider again the independent traffic lights. Let action *switch2green* denote the switching to green. Similarly *switch2red* denotes the switching to red. The fairness assumption

$$\mathcal{F} \;=\; \{\,\{\,switch2green_1, switch2red_1\,\}, \{\,switch2green_2, switch2red_2\,\}\,\}$$

expresses that both traffic lights infinitely often switch color. In this case, it is irrelevant whether strong, weak, or unconditional fairness is required.



Note that in this example $\mathcal{F}$ is *not* a verifiable system property (as it is not guaranteed to hold), but a natural property which is satisfied for a practical implementation of the system (with two independent processors). Obviously,

$$TrLight_1 \;|||\; TrLight_2 \;\models_{\mathcal{F}}\; \text{“each traffic light is green infinitely often”}$$

while the corresponding proposition for the nonfair relation $\models$ is refuted. ■

When using Fairness?

- Fairness may be needed only for liveness constraints
- Irrelevant for safety properties
- Fairness constraints are easy to specify with Buchi Automata (finite automata on infinite words) or with Linear Temporal Logic (LTL)
  - Adding them to transition systems can be more complex.

Formally:

$$Act(s) \;=\; \{\alpha \in Act \mid \exists s' \in S.\, s \xrightarrow{\alpha} s' \,\}.$$

For transition system $TS = (S, Act, \rightarrow, I, AP, L)$ without terminal states, $A \subseteq Act$, and infinite execution fragment $\rho = s_0 \xrightarrow{\alpha_0} s_1 \xrightarrow{\alpha_1} \ldots$ of TS:

1. ρ is *unconditionally A-fair* whenever $\overset{\infty}{\exists}\, j.\ \alpha_j \in A$.

2. ρ is *strongly A-fair* whenever

$$\Big(\overset{\infty}{\exists}\, j.\ Act(s_j) \cap A \neq \varnothing \Big) \implies \Big(\overset{\infty}{\exists}\, j.\ \alpha_j \in A \Big).$$

3. ρ is *weakly A-fair* whenever

$$\Big(\overset{\infty}{\forall}\, j.\ Act(s_j) \cap A \neq \varnothing \Big) \implies \Big(\overset{\infty}{\exists}\, j.\ \alpha_j \in A \Big).$$

$\overset{\infty}{\exists}\, j$ stands for "there are infinitely many j" and $\overset{\infty}{\forall}\, j$ for "for nearly all j"

Fairness Assumption

A *fairness assumption* for Act is a triple

$$\mathcal{F} \;=\; (\mathcal{F}_{ucond}, \mathcal{F}_{strong}, \mathcal{F}_{weak})$$

with $\mathcal{F}_{ucond}, \mathcal{F}_{strong}, \mathcal{F}_{weak} \subseteq 2^{Act}$. Execution ρ is $\mathcal{F}$-fair if

- it is unconditionally A-fair for all $A \in \mathcal{F}_{ucond}$,
- it is strongly A-fair for all $A \in \mathcal{F}_{strong}$, and
- it is weakly A-fair for all $A \in \mathcal{F}_{weak}$.

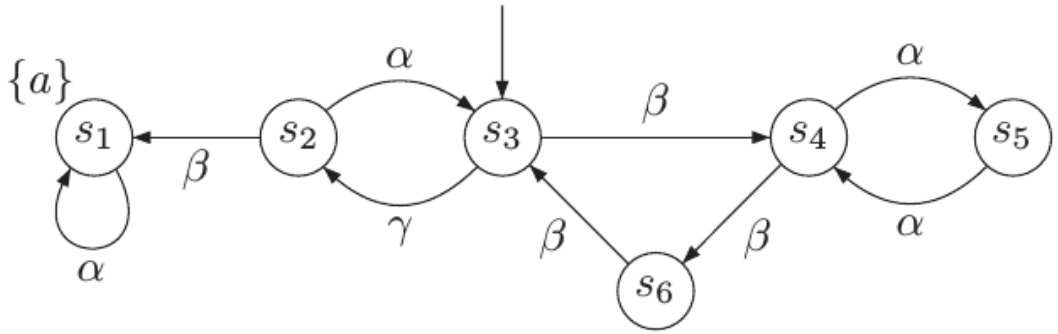
Fair satisfaction

- Traces that satisfy a fairness constraint F are called Fair-Traces.

Let P be an LT property over AP and $\mathcal{F}$ a fairness assumption over Act. Transition system $TS = (S, Act, \rightarrow, I, AP, L)$ *fairly satisfies* P, notation $TS \models_{\mathcal{F}} P$, if and only if $FairTraces_{\mathcal{F}}(TS) \subseteq P$. ■

$$TS \models_{\mathcal{F}_{weak}} P \Rightarrow TS \models_{\mathcal{F}_{strong}} P \Rightarrow TS \models_{\mathcal{F}_{ucond}} P.$$

Example: A TS with AP = {a}



$$\mathcal{F} = (\varnothing, \{\{\alpha\}, \{\beta\}\}, \{\{\beta\}\})$$

Determine whether $TS \models_{\mathcal{F}}$ "eventually a".

- Meaning:
  - no unconditional fairness (set is empty)
  - all transitions labeled with α and all transitions labeled with β have strong fairness requirement
  - transitions labeled with β have weak fairness requirement
- Hint: try to falsify the property: is there a fair trace violating «eventually a»?

Summary: traces, invariants, safety

- ***Trace:* a sequence of *sets* of atomic propositions**. The traces of a transition system TS are obtained from projecting the paths to the sequence of state labels.
- **A linear-time (LT) property is a set of infinite words over alphabet 2^{AP}**
- Two transition systems are **trace-equivalent** (i.e., they exhibit the same traces) if and only if they satisfy the same LT properties.
- An **invariant is an LT property that** is purely state-based and **requires a propositional logic formula Φ to hold for all reachable states**. Invariants can be checked using a depth-first search.
- **Safety properties** are generalizations of invariants. They **constrain the finite behaviors only**. The formal definition of **safety properties can be provided by means of their bad prefixes** in the sense that each trace that refutes a safety property has a finite prefix, the bad prefix, that causes this.
- **Two transition systems exhibit the same finite traces if and only if they satisfy the same safety properties.**

Summary: liveness, fairness

- **A *liveness* property is an LT property if it does not rule out any finite behavior.** It constrains infinite behaviors only.
- Any LT property is equivalent to an LT property that is a conjunction of a safety and a liveness property.
- *Fairness* assumptions serve to rule out traces that are considered to be unrealistic.
- They consist of ***unconditional*, *strong*, and *weak* fairness** constraints, i.e., constraints on the *actions* that occur along infinite executions.

$$TS \models_{\mathcal{F}_{weak}} P \;\Rightarrow\; TS \models_{\mathcal{F}_{strong}} P \;\Rightarrow\; TS \models_{\mathcal{F}_{ucond}} P.$$

- Fairness assumptions are often necessary to establish liveness properties, but they are—provided they are realizable—irrelevant for safety properties.

Addendum: Fairness and Safety

Let TS be a transition system with the set of actions Act and $\mathcal{F}$ a fairness assumption for Act. $\mathcal{F}$ is called *realizable* for TS if for every reachable state s: $FairPaths_{\mathcal{F}}(s) \neq \varnothing$. ■

Theorem 3.55. Realizable Fairness is Irrelevant for Safety Properties

Let TS be a transition system with set of propositions AP, $\mathcal{F}$ a realizable fairness assumption for TS, and P_{safe} a safety property over AP. Then:

$$TS \models P_{safe} \quad \text{if and only if} \quad TS \models_{\mathcal{F}} P_{safe}.$$