Linear Time Properties (3.2.3)

Def 3.10 [LT Property]

A linear-time property (LT property) over the set of atomic propositions AP is a subset of $(2^{AP})^\omega$

set of words that arise from infinite concatenation of symbols from 2^{AP}.

Note: we are only interested in T.S. without terminal states.

Def 3.11 (Satisfaction relation for LT Properties)

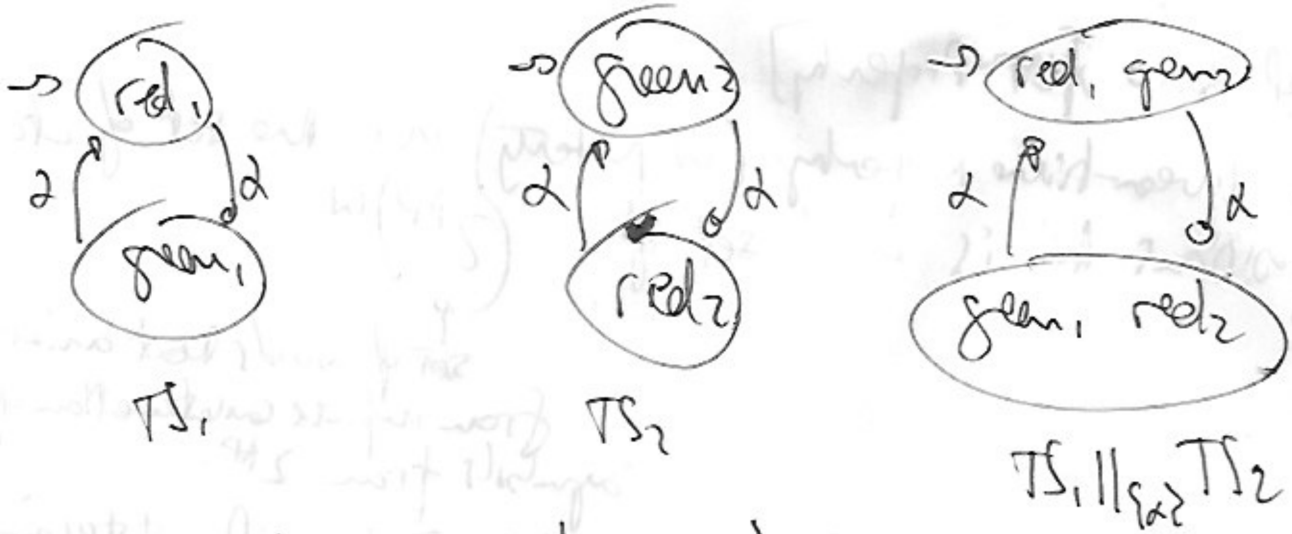
P LT-property over AP

TS a T.S. without terminal states.

$TS \models P$ iff $Traces(TS) \subseteq P$ — all behaviors are admissible.

$s \models P$ whenever $Traces(s) \subseteq P$

Example (3.12) (Note: requires introducing synchronous message passing)



Let $AP = \{red_1, green_1, red_2, green_2\}$

Consider the property P: "The first traffic light is infinitely often green."

This corresponds to words over 2^{AP} of the form

$A_0 A_1 A_2 \ldots$

where $green_1 \in A_i$ holds for infinitely many i.

Eg

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

$\emptyset \{green_1\} \emptyset \{green_1\} \emptyset \{green_1\} \emptyset \ldots$

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

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

MCS - 8

But not

$(red, green_1)(red_1, green_1) \emptyset \emptyset \emptyset \ldots$

eg Property P: "The traffic lights are never both green simultaneously".

Compared to infinite words over 2^{AP}, $A_0 A_1 A_2 \ldots$ s.t. $green_1 \notin A_i$ or $green_2 \notin A_i$, $\forall i \geq 0$.

Example 3.13

$$AP = \{crit_1, crit_2\}$$

$$P_{mutex} = A_0A_1A_2\ldots \in (2^{AP})^\omega \text{ s.t. } \{crit_1, crit_2\} \not\subseteq A_i,\ \forall i \geq 0.$$

Ex. 3.14 $\quad AP = \{wait_1, crit_1, wait_2, crit_2\}$

$$P_{finwait} = A_0A_1A_2\ldots \in (2^{AP})^\omega \text{ s.t.}$$

$$\forall j.\ wait_i \in A_j \Rightarrow \exists k > j.\ crit_i \in A_k \quad \forall i \in 1..2$$

Each of the two processes enters its CS eventually if they are waiting

$$P_{nostarve} = A_0A_1A_2\ldots \in (2^{AP})^\omega \text{ s.t.}$$

$$\left(\forall k \geq 0.\ \exists j \geq k.\ wait_i \in A_j\right) \Rightarrow \left(\forall k \geq 0.\ \exists j \geq k.\ crit_i \in A_j\right) \quad i \in \{1..2\}$$

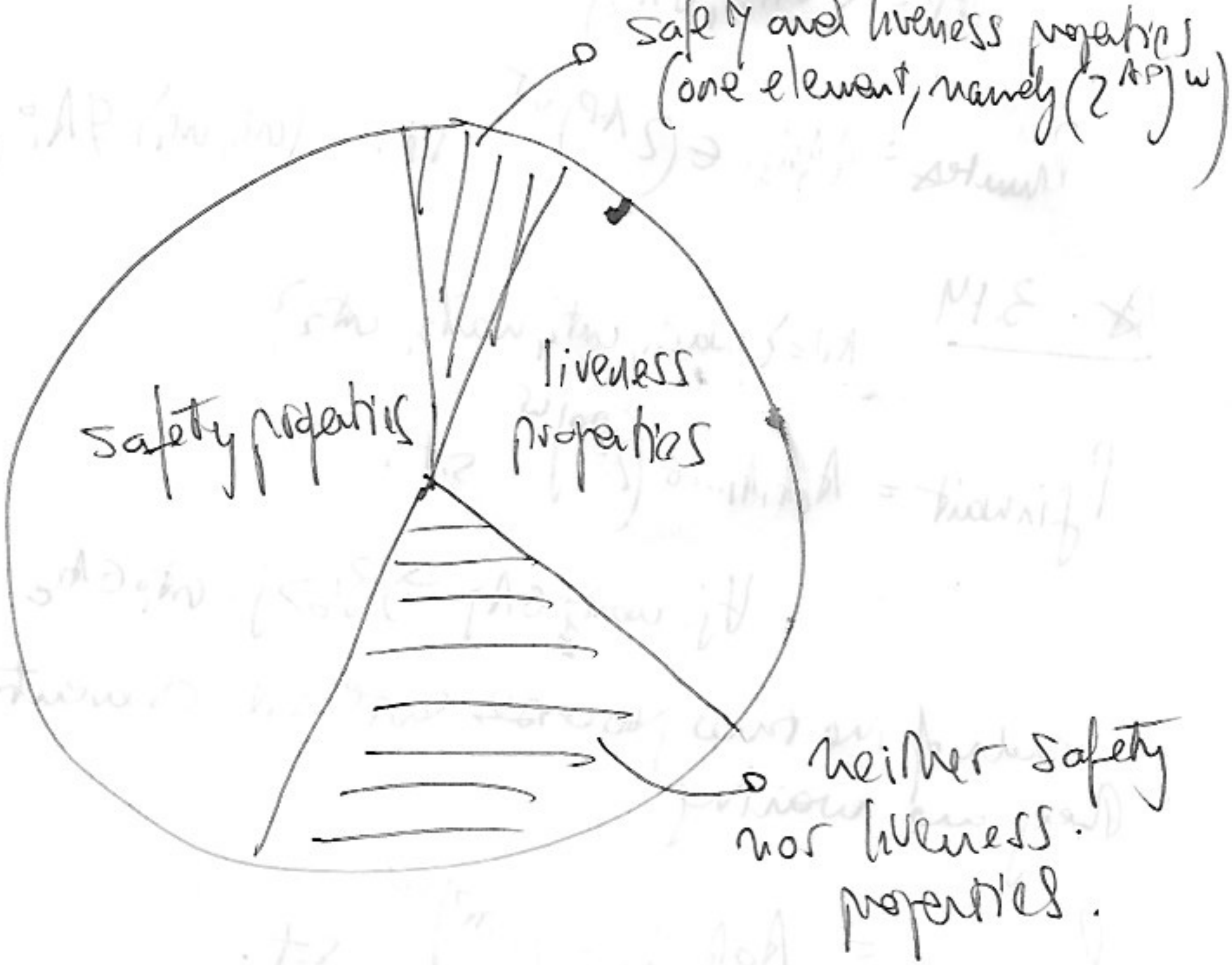
In abbreviated form.

$$\left(\overset{\infty}{\exists} j.\ wait_i \in A_j\right) \Rightarrow \left(\overset{\infty}{\exists} j.\ crit_i \in A_j\right)$$

Failure of starvation for semaphore-based example:

$$\emptyset\ \left(\{wait_2\}\ \{wait_1, wait_2\}\ \{crit_1, wait_2\}\right)^\omega$$

Classification of LT properties (3.4.2)



Safety properties (3.3.2)

Def (3.22)

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 $\hat{\sigma}$ of σ s.t.

$$P_{safe} \cap \{\sigma' \in (2^{AP})^\omega \mid \hat{\sigma} \text{ is a finite prefix of } \sigma'\} = \emptyset$$

Any such finite word $\hat{\sigma}$ is called a bad prefix.

Ex. Traffic light.

"Always at least one of the lights is on"

$$\{\sigma = A_0 A_1 \dots \mid A_j \subseteq AP \wedge A_j \neq \emptyset\}$$

bad prefixes:

finite words that contain $\emptyset$.

"It is never the case that two lights are switched on at the same time"

$$\{\sigma = A_0 \dots \mid A_j \subseteq AP \wedge |A_j| \leq 1\}$$

Bad prefixes for this property are words containing sets such as $\{red, green\}$, $\{red, yellow\}$ and so on.

Minimal bad prefixes end with such sets.

Let $AP' = \{red, yellow\}$. Consider the property "a red phase must be preceded immediately by a yellow phase". This property is specified by

$$\{\sigma = A_0 A_1 \ldots \mid A_i \subseteq \{red, yellow\} \wedge \forall i \geq 0. red \in A_i \Rightarrow yellow \in A_{i-1}\}$$

Examples of bad prefixes are

$$\emptyset \emptyset \{red\} \quad \text{and} \quad \emptyset \{red\}$$

The following bad prefix is not minimal

$$\underbrace{\{yellow\}\{yellow\}\{red\}\{red\}}_{\text{also a bad prefix.}}\emptyset\{red\}$$

Consider the property

"a red light eventually turns on"

This property is specified by

$$\{\sigma = A_0 A_1 \ldots \mid A_i \subseteq \{red, yellow\} \wedge \exists j \geq 0 . red \in A_j\}$$

Take $\sigma = \emptyset^\omega \in (2^{AP})^\omega \setminus P$

All finite prefixes of σ are of the form $\emptyset^n$.

Note that

$$P_{safe} \cap \underbrace{\{\sigma' \in (2^{AP})^\omega \mid \emptyset^n \text{ is a finite prefix of } \sigma'\}}_{\text{take, for example, } \emptyset^n\{red\}} \neq \emptyset$$

Hence P is not a safety property.

Eg. 3.24.

Consider the vending machine and the property

"the number of coins is always at least the number of dispensed drinks".

$$\{\sigma = A_0 \ldots \mid \forall i \geq 0.\ |\{0 \le j \le i \mid pay \in A_j\}| \ge |\{0 \le j \le i \mid drink \in A_j\}|\}$$

Bad prefixes for this property are

$\emptyset\ \{pay\}\ \{drink\}\ \{drink\}$

$\emptyset\ \{pay\}\ \{drink\}\ \emptyset\ \{pay\}\ \{drink\}\ \{drink\}$

Liveness Properties (3.4)

Sometimes called "progress" properties: "something good" will happen in the future.

whereas safety properties are violated in finite time, i.e. by a finite system run, liveness properties are violated in infinite time, i.e, by infinite system runs.

Liveness properties require a certain condition on the infinite behaviors.

Def. 3.33 (Liveness Property)

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 s.t. each finite word can be extended to an infinite word that satisfies P.

$$\forall w \in (2^{AP})^* \; \exists \sigma \in (2^{AP})^\omega . \; w\sigma \in P$$

Examples (3.34)

a) (eventually) Each process will eventually enter its critical section

b) (repeatedly eventually) each process will enter its critical section infinitely often.

c) (starvation freedom) each waiting process will eventually enter its critical section.

$AP = \{wait_1, crit_1, wait_2, crit_2\}$

All $A_0 A_1 \ldots \in (2^{AP})^\omega$ s.t.

a) $\left(\exists j \geq 0.\ crit_1 \in A_j\right) \wedge \left(\exists j \geq 0.\ crit_2 \in A_j\right)$

b)
$$\left(\forall k \geq 0.\ \exists j > k.\ crit_1 \in A_j\right)$$
$$\wedge$$
$$\left(\forall k \geq 0.\ \exists j > k.\ crit_2 \in A_j\right)$$

Abbreviation:
$$\left(\overset{\infty}{\exists} j \geq 0.\ crit_1 \in A_j\right) \wedge \left(\overset{\infty}{\exists} j \geq 0.\ crit_2 \in A_j\right)$$

c)
$$\forall j \geq 0. \left(\left(wait_1 \in A_j\right) \Rightarrow \left(\exists k > j.\ crit_1 \in A_k\right)\right)$$
$$\wedge$$
$$\forall j \geq 0 \left(\left(wait_2 \in A_j\right) \Rightarrow \left(\exists k > j.\ crit_2 \in A_k\right)\right).$$

Note: we assume that a process that starts waiting to acquire access to the c.s. does not "give up".