4. Linear-Time Properties

Computer-Aided Verification

Dave Parker

University of Birmingham

2017/18

Announcements

- Continuous assessment
  - reminder: 4 assignments (5 for “extended” version)
  - due Thurs of weeks 3, 5, 8, 11 (and week 10 for extra one)
- Assignment 1 (models and properties)
  - formative; out now; due 12 noon Thur 25 Jan
  - submitted through Canvas
  - solutions worked through in tutorial sessions
- Next week
  - Thur lecture is moved to the tutorial slot:
  - Fri 10am (SportEx Lecture Theatre 1)

Recap: Modelling

- Nondeterminism
  - multiple possible behaviours of system being modelled
  - uses: unknown environments/inputs, abstraction, concurrency

- Parallel composition – key ideas:

- Nondeterminism models interleaving of parallel components
  - i.e., unknown execution order (or unknown scheduling)

- Parallel composition requires states of both components
  - i.e., resulting LTS has product state space $S_1 \times S_2$

Today

- Linear-time properties
  - formal definition
  - paths, traces, satisfaction

- Important classes of properties
  - invariants
  - safety
  - liveness

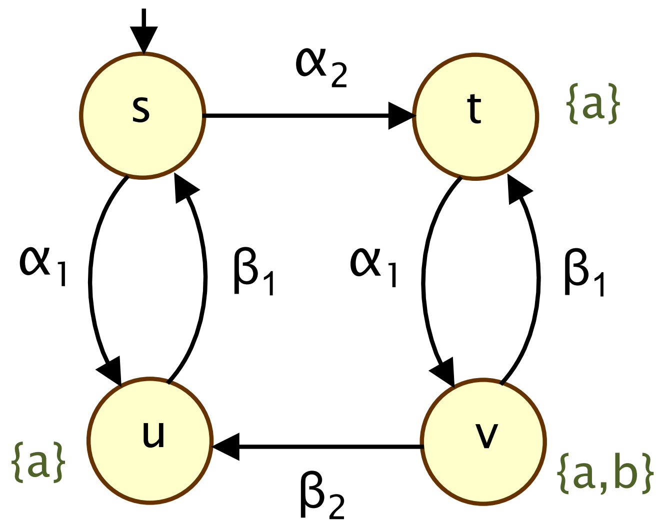
- See [BK08] chapter 3 (specifically: 3.2–3.4)

Some assumptions

- We will assume LTSs are finite
  - since we start to consider algorithms to check properties
- We assume no deadlocks
  - i.e. LTSs have no terminal states
  - and so all maximal paths are infinite
  - (we can easily check for deadlocks and "repair" them)

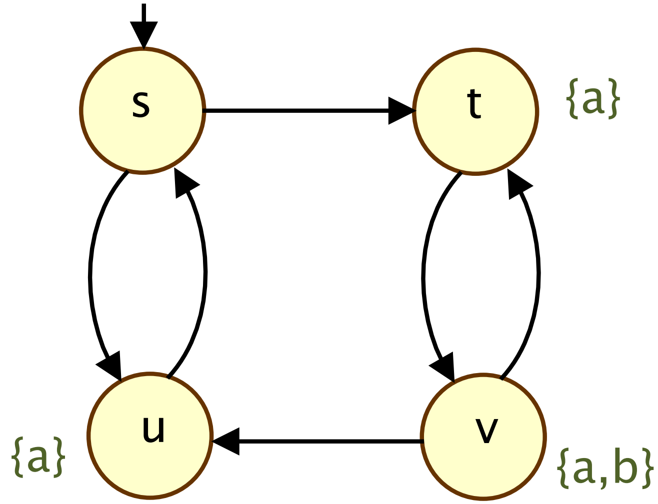
LTS labels

- Recall:
  - state labels (atomic prop.s) are used for facts/observations
  - transition labels (actions) primarily for interaction/composition



LTS labels

- Recall:
  - state labels (atomic prop.s) are used for facts/observations
  - transition labels (actions) primarily for interaction/composition

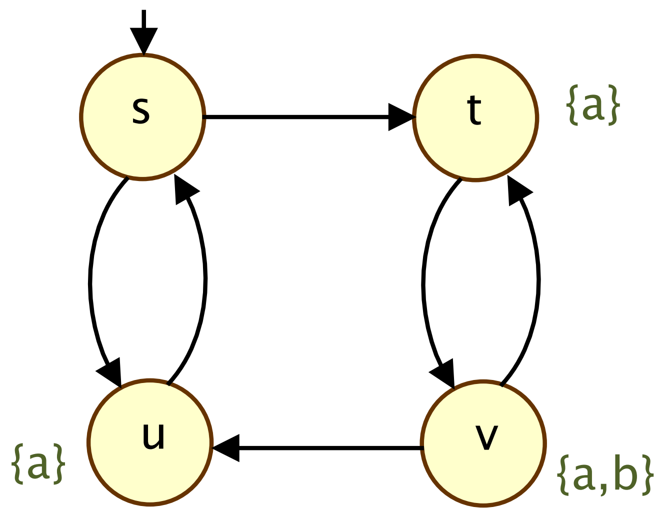


- So:
  1. properties are formally expressed using atomic propositions
  2. technically, can work on underlying graph of an LTS

- Paths
  - are now of the form $\pi = s\ t\ v\ t\ v\ u \ldots$
  - i.e. we ignore actions

Traces

- Recall:
  - an LTS is a tuple $M = (S, Act, \rightarrow, I, AP, L)$
  - with a labelling function $L : S \rightarrow 2^{AP}$
- Example:
  - $AP = \{a,b\}$
  - $2^{AP} = \{\varnothing, \{a\}, \{b\}, \{a,b\}\}$
  - e.g. $L(v) = \{a,b\}$

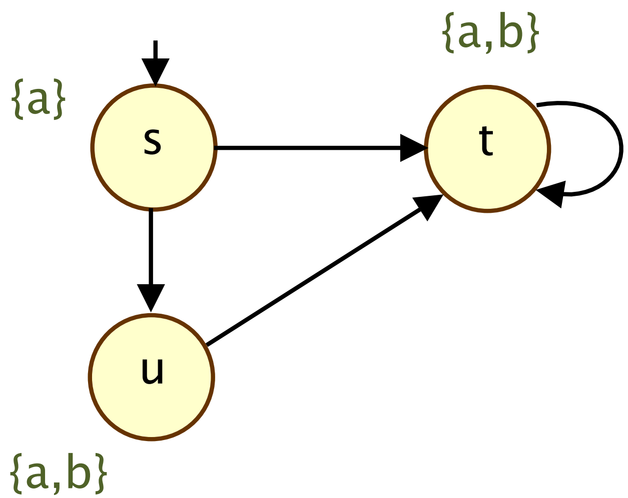


- Traces
  - the sequences of (sets of) atomic propositions true in each state
  - the trace of path $\pi = s_0 s_1 s_2 s_3 \ldots$
  - is $\mathrm{trace}(\pi) = L(s_0)L(s_1)L(s_2)L(s_3)\ldots$
  - e.g. trace(s t v t v u…) = $\varnothing$ {a} {a,b} {a} {a,b} {a} …

Notation: Paths and traces

- For LTS $M = (S, Act, \rightarrow, I, AP, L)$:
  - Paths(M) is the set of all paths starting from an initial state in I
  - Traces(M) is the set for all traces of those paths

- Example

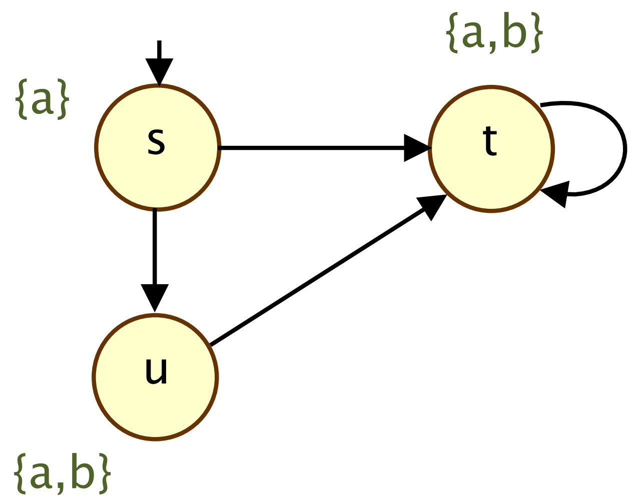


  - Paths(M) = { s u t^ω, s t^ω }
  - Traces(M) = { {a} $\{a,b\}^\omega$ }

Linear-time properties

- A linear-time property is
  - (informally) a set of traces that an LTS is allowed to exhibit
  - e.g. "b appears at most once", "a and b never appear together"
  - (formally) a subset of $(2^{AP})^{\omega}$, i.e., a language of infinite words

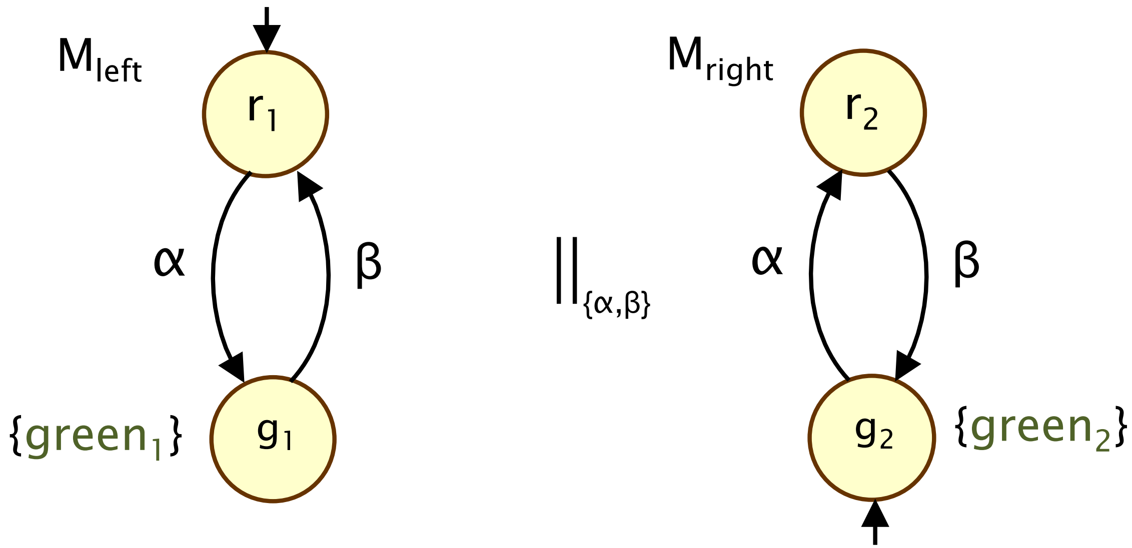
- Satisfaction: $M \vDash P$
  - of a property $P \subseteq (2^{AP})^{\omega}$ by an LTS M
  - we say "M satisfies P", or "P is true in M"
  - defined as: $M \vDash P \;\Leftrightarrow\; Traces(M) \subseteq P$



- Note:
  - properties are <u>not</u> tied to a particular model
  - we sometimes specify the complement of P (“good” vs. ”bad”)

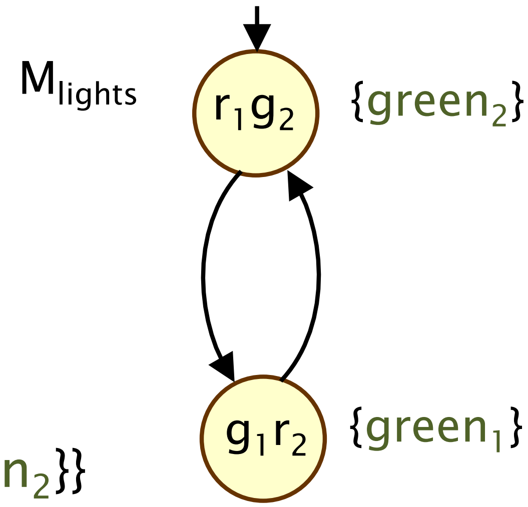
Example

- Example: a pair of traffic lights
  - $M_{lights} = M_{left} \,||_{\{\alpha,\beta\}}\, M_{right}$



Example

- Example: a pair of traffic lights
  - $M_{lights} = M_{left} \,||_{\{\alpha,\beta\}}\, M_{right}$
- Labels
  - $AP = \{green_1, green_2\}$
  - $2^{AP} = \{\emptyset, \{green_1\}, \{green_2\}, \{green_1, green_2\}\}$
  - M_{lights} exhibits a single trace



- How do we define this property?
  - P: "the traffic lights never both show green simultaneously"
  - $P = \{\ \{green_2\}\,\{green_1\}\,\{green_2\}\,\{green_1\}\ \dots\ \}$?
  - no, because e.g. $\{green_2\}\,\{green_2\}\,\{green_2\}\ \dots$ is in P
  - properties are not tied to specific models
  - $P = \{\ A_0A_1A_2\ldots \in (2^{AP})^\omega \mid A_j \neq \{green_1, green_2\} \text{ for all } j \geq 0\ \}$

Classes of linear-time properties

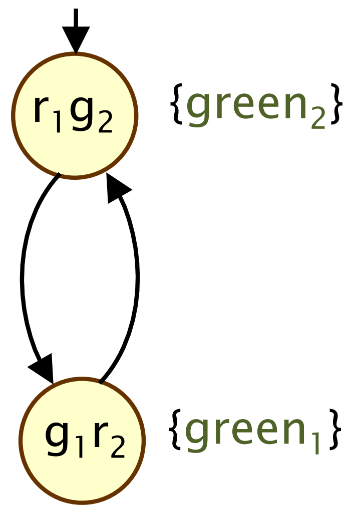
- We identify several useful classes of property
  - important consequences for what properties we can express
  - and what algorithms/techniques are required to verify them

- Defined informally…

- Invariants
  - "something good is always true”
- Safety properties
  - "nothing bad happens”
- Liveness properties
  - "something good happens in the long run"

Invariants

- Informally:
  - a condition Φ about states must always be true
- Formally:
  - $P_{inv} \subseteq (2^{AP})^\omega$ is an invariant if there is a propositional logic formula Φ such that:
  - $P_{inv} = \{ A_0A_1A_2\ldots \in (2^{AP})^\omega \mid A_j \models \Phi \text{ for all } j \geq 0 \}$



- Examples:
  - P_1 = "one of the green lights is always on"
  - $\Phi_1 = green_1 \vee green_2$
  - P_2 = "the traffic lights never both show green simultaneously"
  - $\Phi_2 = \neg(green_1 \wedge green_2)$

Checking invariants

- Invariants:
  - checking invariants can done via reachability
  - $L(s) \vDash \Phi$ for all states s on all paths of the LTS
  - $L(s) \vDash \Phi$ for all reachable states s of the LTS

- Since we assume (for now) LTSs are finite
  - standard graph traversal, e.g. depth-first/breadth-first search
  - identify all reachable states s and check that $L(s) \vDash \Phi$

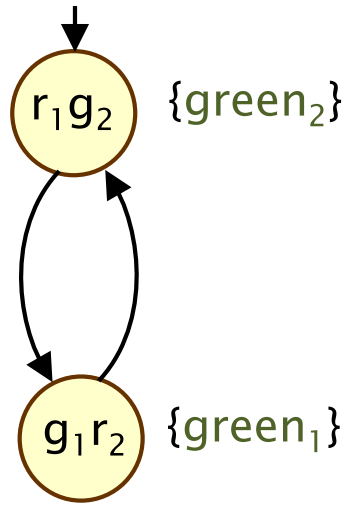
- Improvements
  - stop as soon as a violating state is found (i.e. $L(s) \nvDash \Phi$)
  - use breadth-first search with a stack
    and return a path to the violating state

Safety properties

- Informally:
  - defined in terms of “bad” events, e.g. "a failure does not occur”
  - “bad” events happen in finite time, and cannot be recovered from
- More precisely
  - P_{safe} is a safety property if any (infinite) word where P_{safe} does not hold has a bad prefix
  - a bad prefix is a finite prefix σ' containing the bad event, such that no infinite path beginning with σ' satisfies P_{safe}
- Formally:
  - $P_{safe} \subseteq (2^{AP})^\omega$ is a safety property if, for all words $\sigma \in (2^{AP})^\omega \setminus P_{safe}$, there is a finite prefix σ' of σ such that:
  - $P_{safe} \cap \{ \sigma'' \in (2^{AP})^\omega \mid \sigma' \text{ is a prefix of } \sigma'' \} = \varnothing$

Examples

- All invariants are safety properties
  - e.g. P_2 = "the traffic lights never both show green simultaneously": $\Phi_2 = \neg(green_1 \wedge green_2)$
  - what are the bad prefixes?
  - e.g. {$green_2$} {$green_1$,$green_2$}
  - words of the form $A_0A_1 \ldots A_n$ with $A_i \models \Phi_2$ for all $0 \leq i < n$ and $A_n \not\models \Phi_2$



- But not all safety properties are invariants
  - e.g. P_3 = "$green_1$ is always preceded by $green_2$"
  - what are the bad prefixes?
  - e.g. $\varnothing$ {$green_1$}
  - any word where $green_1$ appears before $green_2$
  - why is this not an invariant?

Question

- Are these safety properties? (assume AP = {$green_1$, $green_2$})
- And why?

  - "at least one of the traffic lights always shows green"

    yes, because it is an invariant, because…
  - "$green_1$ and $green_2$ occur in strict alternation"

    yes, because…
  - $green_2$ is eventually true

    no, because…
  - $green_2$ is true infinitely often

    no because…

- The last two are liveness properties

Liveness properties

- Informally:
  - "something good happens eventually, or in the long run"
  - e.g. "the program always eventually terminates"

- More precisely
  - P_{live} is a liveness property if it does not rule out any prefixes
  - any finite word can be extended to an infinite word in P_{live}

- Formally:
  - $P_{live} \subseteq (2^{AP})^\omega$ is a liveness property if,
    for all finite words $\sigma \in (2^{AP})^*$, there exists
    an infinite word $\sigma' \in (2^{AP})^\omega$ such that $\sigma\,\sigma' \in P_{live}$

Summary

- Paths, traces
  - path: infinite sequence π of states from LTS M
  - trace: infinite word σ over 2^{AP}
- Properties
  - linear-time property = set P of infinite words over 2^{AP}
  - satisfaction: $M \models P$ if all traces of M are in P
- Classes of property
  - invariant: formula Φ is true in all (reachable) states
  - safety property: "nothing bad happens"
    - violating paths have a finite bad prefix
  - liveness: "something good happens in the long run"
    - any finite path can be extended to a satisfying one

Next lecture

- Linear temporal logic
  - see Chapter 5 of [BK08]