Modelling and Validation of Concurrent System

Computational Tree Logic

António Ravara

May 9, 2024

Limitations of HML

Origins of Logic in Computer Science

The role of Logics in Program Verification

Since the 1960s that Computer Scientists have developed logics to help checking program correctness:

- Floyd-Hoare was the first, to state and prove properties of imperative sequential programs using proof systems;

Origins of Logic in Computer Science

The role of Logics in Program Verification

Since the 1960s that Computer Scientists have developed logics to help checking program correctness:

- Floyd-Hoare was the first, to state and prove properties of imperative sequential programs using proof systems;
- Manna and Pnueli developed an approach to check property satisfiability in First-Order Logic (FOL).
  Since stating program properties in FOL is cumbersome, Manna and Pnueli developed *Temporal Logic*, to better express safety and liveness properties.

The role of Logics in Program Verification

Since the 1960s that Computer Scientists have developed logics to help checking program correctness:

- Floyd-Hoare was the first, to state and prove properties of imperative sequential programs using proof systems;
- Manna and Pnueli developed an approach to check property satisfiability in First-Order Logic (FOL).
  Since stating program properties in FOL is cumbersome, Manna and Pnueli developed *Temporal Logic*, to better express safety and liveness properties.
  TLA is a temporal Logic.

The purpose of HML

HML: yet another program logic?

It's purpose is rather different.

Instead of aiming at stating and proving properties of programs, it aims at providing a logical account of the operational equivalence notion of process languages (observational congruence).

The purpose of HML

HML: yet another program logic?

It's purpose is rather different.

Instead of aiming at stating and proving properties of programs, it aims at providing a logical account of the operational equivalence notion of process languages (observational congruence).

- Strong Bisimulation coincides with logical equivalence, considering strong modalities.
- Observational congruence coincides with logical equivalence for finite branch processes, considering weak modalities.

Temporal Properties of Concurrent/Reactive Systems

Assume GOOD (respectively BAD) is a(n un)desired property of a (distributed) system.

Temporal Properties of Concurrent/Reactive Systems

Assume GOOD (respectively BAD) is a(n un)desired property of a (distributed) system.

Temporal Logics - it's about behaviour *over time*

- It is never the case that something bad happens.
  (strong) *Safety*: for all reachable states, $\neg$BAD holds.

Temporal Properties of Concurrent/Reactive Systems

Assume GOOD (respectively BAD) is a(n un)desired property of a (distributed) system.

Temporal Logics - it's about behaviour *over time*

- It is never the case that something bad happens.
  (strong) *Safety*: for all reachable states, $\neg$BAD holds.
- Eventually, something good will happen.
  (weak) *Liveness*: for some reachable states, GOOD holds.

Temporal Properties of Concurrent/Reactive Systems

Assume GOOD (respectively BAD) is a(n un)desired property of a (distributed) system.

Temporal Logics - it's about behaviour *over time*

- It is never the case that something bad happens.
  (strong) *Safety*: for all reachable states, $\neg$BAD holds.
- Eventually, something good will happen.
  (weak) *Liveness*: for some reachable states, GOOD holds.

Can one define such properties in HML?

Temporal Properties of Concurrent/Reactive Systems

Assume GOOD (respectively BAD) is a(n un)desired property of a (distributed) system.

Temporal Logics - it's about behaviour *over time*

- It is never the case that something bad happens.
  (strong) *Safety*: for all reachable states, $\neg$BAD holds.
- Eventually, something good will happen.
  (weak) *Liveness*: for some reachable states, GOOD holds.

Can one define such properties in HML?

There is no way of talking about something holding eventually or always in the future

Assume GOOD (respectively BAD) is a(n un)desired property of a (distributed) system.

Temporal Logics - it's about behaviour *over time*

- It is never the case that something bad happens.
  (strong) *Safety*: for all reachable states, $\neg$BAD holds.
- Eventually, something good will happen.
  (weak) *Liveness*: for some reachable states, GOOD holds.

Can one define such properties in HML?

There is no way of talking about something holding eventually or always in the future — formulæ in HML talk about a finite number of steps, the concrete number given by the depth of a formula.

Computational Tree Logic, CTL

A simple logic to express temporal properties of processes

In the early 1980s, Clarke, Emerson, Sistla and others developed a logic with a *next state* modality and built-in temporal connectives

Computational Tree Logic, CTL

It features:

- propositional connectives
- a one-step possibility modality
- a temporal operator: *until*:
  φ holds *until* ψ holds

A simple logic to express temporal properties of processes

In the early 1980s, Clarke, Emerson, Sistla and others developed a logic with a *next state* modality and built-in temporal connectives

Computational Tree Logic, CTL

It features:

- propositional connectives
- a one-step possibility modality
- a temporal operator: *until*:
  φ holds *until* ψ holds

The key idea

- Quantify over *paths* on the LTS
  (instead of only over actions)
- Talk about the future

The *until* temporal operator

Like in HML, CTL formulæ are interpreted over LTSs.

A *path* on an LTS S

is a sequence of states, linked to each other by transitions.

If P is an initial state of S, then

$$P \xrightarrow{a_1} P_1 \xrightarrow{a_2} P_2 \cdots$$

is a path of S if each triple $P_{i-1} \xrightarrow{a_i} P_i$ is in S's transition relation

The *until* temporal operator

Like in HML, CTL formulæ are interpreted over LTSs.

A *path* on an LTS S

is a sequence of states, linked to each other by transitions.

If P is an initial state of S, then

$$P \xrightarrow{a_1} P_1 \xrightarrow{a_2} P_2 \cdots$$

is a path of S if each triple $P_{i-1} \xrightarrow{a_i} P_i$ is in S's transition relation

Paths may be finite or infinite.

The *until* temporal operator

Like in HML, CTL formulæ are interpreted over LTSs.

A *path* on an LTS S

is a sequence of states, linked to each other by transitions.

If P is an initial state of S, then

$$P \xrightarrow{a_1} P_1 \xrightarrow{a_2} P_2 \cdots$$

is a path of S if each triple $P_{i-1} \xrightarrow{a_i} P_i$ is in S's transition relation

Paths may be finite or infinite.

Computational Tree

The acyclic unfolding of an LTS

Syntax of CTL

Consider $\mathcal{K} \subseteq \mathsf{Act}$

The set $\mathcal{F}$ of formulæ is inductively defined by the grammar

$$\varphi ::= \bot \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid \langle\mathcal{K}\rangle\varphi \mid \mathtt{E}(\varphi\,\mathtt{U}\,\psi) \mid \mathtt{A}(\varphi\,\mathtt{U}\,\psi)$$

where the temporal operators:

- $\mathtt{E}$ means *there Exists* a path in the LTS
  path *p* satisfies $\varphi\,\mathtt{U}\,\psi$ if it has a state that satisfies ψ and all previous states satisfy φ
- $\mathtt{A}$ means *in All* paths

Syntax of CTL

Consider $\mathcal{K} \subseteq \mathsf{Act}$

The set $\mathcal{F}$ of formulæ is inductively defined by the grammar

$$\varphi ::= \bot \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid \langle\mathcal{K}\rangle\varphi \mid \mathtt{E}(\varphi\,\mathtt{U}\,\psi) \mid \mathtt{A}(\varphi\,\mathtt{U}\,\psi)$$

where the temporal operators:

- $\mathtt{E}$ means *there Exists* a path in the LTS
  path *p* satisfies $\varphi\,\mathtt{U}\,\psi$ if it has a state that satisfies ψ and all previous states satisfy φ
- $\mathtt{A}$ means *in All* paths

Furthermore, some derived temporal operators are used often

- $\mathtt{F}$ means *sometime* in the Future
  $\mathtt{F}\,\varphi = \top\,\mathtt{U}\,\varphi$
- $\mathtt{G}$ means *always* in the future
  $\mathtt{G}\,\varphi = \neg(\top\,\mathtt{U}\,\neg\varphi)$

Semantics of CTL

The *satisfaction relation*:

$\models \subseteq \mathrm{Proc} \times \mathcal{F}$ **is defined as for HML**

$P_0 \models \mathtt{E}(\varphi \,\mathtt{U}\, \psi)$ if for SOME path $P_0 \xrightarrow{a_1} P_1 \xrightarrow{a_2} \cdots$ it is the case that $\exists\, i\,.\,(i \geq 0 \land P_i \models \psi \land \forall\, j\,.\,(0 \leq j < i \land P_j \models \varphi)$

$P_0 \models \mathtt{A}(\varphi \,\mathtt{U}\, \psi)$ if for ALL paths $P_0 \xrightarrow{a_1} P_1 \xrightarrow{a_2} \cdots$ it is the case that $\exists\, i\,.\,(i \geq 0 \land P_i \models \psi \land \forall\, j\,.\,(0 \leq j < i \land P_j \models \varphi)$

Semantics of CTL

The *satisfaction relation*:

$\models \subseteq \mathrm{Proc} \times \mathcal{F}$ **is defined as for HML**

$P_0 \models \mathtt{E}(\varphi\,\mathtt{U}\,\psi)$ if for SOME path $P_0 \xrightarrow{a_1} P_1 \xrightarrow{a_2} \cdots$ it is the case that $\exists\, i\,.\,(i \geq 0 \wedge P_i \models \psi \wedge \forall\, j\,.\,(0 \leq j < i \wedge P_j \models \varphi)$

$P_0 \models \mathtt{A}(\varphi\,\mathtt{U}\,\psi)$ if for ALL paths $P_0 \xrightarrow{a_1} P_1 \xrightarrow{a_2} \cdots$ it is the case that $\exists\, i\,.\,(i \geq 0 \wedge P_i \models \psi \wedge \forall\, j\,.\,(0 \leq j < i \wedge P_j \models \varphi)$

More useful abbreviations

- $\mathtt{E}\,\mathtt{F}\,\varphi = \mathtt{E}\,(\top\,\mathtt{U}\,\varphi)$
- $\mathtt{A}\,\mathtt{F}\,\varphi = \mathtt{A}\,(\top\,\mathtt{U}\,\varphi)$
- $\mathtt{E}\,\mathtt{G}\,\varphi = \neg\mathtt{A}\,\mathtt{F}\,\neg\varphi$
- $\mathtt{A}\,\mathtt{G}\,\varphi = \neg\mathtt{E}\,\mathtt{F}\,\neg\varphi$

Semantics of CTL

The *satisfaction relation*:
$\models \; \subseteq \text{Proc} \times \mathcal{F}$ **is defined as for HML**

$P_0 \models \mathtt{E}(\varphi \, \mathtt{U} \, \psi)$ if for SOME path $P_0 \xrightarrow{a_1} P_1 \xrightarrow{a_2} \cdots$ it is the case that $\exists\, i \,.\, (i \geq 0 \wedge P_i \models \psi \wedge \forall\, j \,.\, (0 \leq j < i \wedge P_j \models \varphi)$

$P_0 \models \mathtt{A}(\varphi \, \mathtt{U} \, \psi)$ if for ALL paths $P_0 \xrightarrow{a_1} P_1 \xrightarrow{a_2} \cdots$ it is the case that $\exists\, i \,.\, (i \geq 0 \wedge P_i \models \psi \wedge \forall\, j \,.\, (0 \leq j < i \wedge P_j \models \varphi)$

More useful abbreviations

- $\mathtt{E}\,\mathtt{F}\,\varphi = \mathtt{E}\,(\top\,\mathtt{U}\,\varphi)$
- $\mathtt{A}\,\mathtt{F}\,\varphi = \mathtt{A}\,(\top\,\mathtt{U}\,\varphi)$
- $\mathtt{E}\,\mathtt{G}\,\varphi = \neg \mathtt{A}\,\mathtt{F}\,\neg\varphi$
- $\mathtt{A}\,\mathtt{G}\,\varphi = \neg \mathtt{E}\,\mathtt{F}\,\neg\varphi$

In CTL temporal operators are always prefixed by path quantifiers.

Properties of Distributed Systems in CTL

Strong Safety $\mathtt{A}\,\mathtt{G}\,\neg \mathrm{BAD}$

Strong Liveness $\mathtt{A}\,\mathtt{F}\,\mathrm{GOOD}$

Weak Safety $\mathtt{E}\,\mathtt{G}\,\neg \mathrm{BAD}$

Weak Liveness $\mathtt{E}\,\mathtt{F}\,\mathrm{GOOD}$

Assume two processes willing to access a critical region.

Absence of Deadlock

Every path is infinite: $\mathtt{A}\,\mathtt{G}\,\langle - \rangle \top$

Properties of Distributed Systems in CTL

Strong Safety $\mathtt{A}\,\mathtt{G}\,\neg \mathsf{BAD}$

Strong Liveness $\mathtt{A}\,\mathtt{F}\,\mathsf{GOOD}$

Weak Safety $\mathtt{E}\,\mathtt{G}\,\neg \mathsf{BAD}$

Weak Liveness $\mathtt{E}\,\mathtt{F}\,\mathsf{GOOD}$

Assume two processes willing to access a critical region.

Absence of Deadlock

Every path is infinite: $\mathtt{A}\,\mathtt{G}\,\langle - \rangle \top$

Absence of Starvation

In every path, a process wishing to enter the critical region will eventually do $\mathtt{A}\,\mathtt{G}\,[\mathit{acq}1]\mathtt{A}\,\mathtt{F}\,\langle \mathit{rel}1 \rangle \top$

Properties of Distributed Systems in CTL

Strong Safety $\mathtt{A}\,\mathtt{G}\,\neg\mathrm{BAD}$

Strong Liveness $\mathtt{A}\,\mathtt{F}\,\mathrm{GOOD}$

Weak Safety $\mathtt{E}\,\mathtt{G}\,\neg\mathrm{BAD}$

Weak Liveness $\mathtt{E}\,\mathtt{F}\,\mathrm{GOOD}$

Assume two processes willing to access a critical region.

Absence of Deadlock

Every path is infinite: $\mathtt{A}\,\mathtt{G}\,\langle - \rangle \top$

Absence of Starvation

In every path, a process wishing to enter the critical region will eventually do $\mathtt{A}\,\mathtt{G}\,[\mathit{acq}1]\mathtt{A}\,\mathtt{F}\,\langle \mathit{rel}1 \rangle \top$

Mutual exclusion

In no path are two processes simultaneously in the critical region $\mathtt{A}\,\mathtt{G}\,([\mathit{rel}1]\bot \vee [\mathit{rel}2]\bot)$

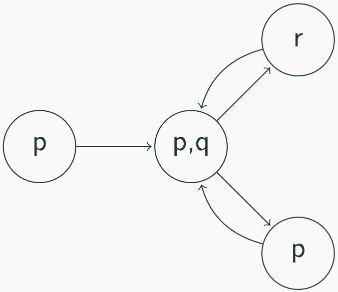
An LTS and some properties

Consider the LTS below. To simplify we omit the actions and just state as propositions the properties holding in each state.

An LTS and some properties

Consider the LTS below. To simplify we omit the actions and just state as propositions the properties holding in each state.

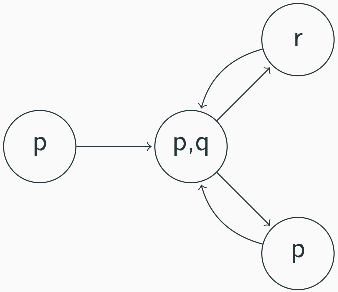
Call P to the leftmost state, Q to the central one, R to the top-right one, and S to the bottom-right one.



An LTS and some properties

Consider the LTS below. To simplify we omit the actions and just state as propositions the properties holding in each state.

Call P to the leftmost state, Q to the central one, R to the top-right one, and S to the bottom-right one.



- $P \models \mathtt{A}(p\,\mathtt{U}\,q)$ does not hold.
- $P \models \mathtt{E}\,\mathtt{F}\,r$ holds
- $P \models \mathtt{A}\,\mathtt{F}\,r$ does not hold.
- $P \models \mathtt{E}\,\mathtt{F}\,p$ holds
- $P \models \mathtt{A}\,\mathtt{F}\,(\mathtt{E}\,\mathtt{G}\,r \vee \mathtt{E}\,\mathtt{G}\,p)$ does not hold.

Beyond CCS

Limitations of CCS
The communication topology is rigid, not allowing to model, *e.g.*, a mobile phone network.

Beyond CCS

Limitations of CCS
The communication topology is rigid, not allowing to model, *e.g.*, a mobile phone network.

Solution
Allow to pass (newly created) names (instead of only values)

The π-calculus – supports making new acquaintances and breaking links

Beyond CCS

Limitations of CCS
The communication topology is rigid, not allowing to model, *e.g.*, a mobile phone network.

Solution
Allow to pass (newly created) names (instead of only values)

The π-calculus – supports making new acquaintances and breaking links

Beyond the π-calculus

- Higher-order π allows to pass processes
- SPI adds encryption and decryption primitives
- ψ-calculi allows to have terms of an algebra instead of names

Limitations of CTL

It does not allow free alternation of temporal and modal operators.

Pure temporal properties, for example, are not expressible.

Beyond CTL

Limitations of CTL

It does not allow free alternation of temporal and modal operators.

Pure temporal properties, for example, are not expressible.

Solutions

- CTL* lifts the alternation limitation

Beyond CTL

Limitations of CTL

It does not allow free alternation of temporal and modal operators.

Pure temporal properties, for example, are not expressible.

Solutions

- CTL* lifts the alternation limitation
- The modal μ-calculus substitutes the temporal operators by fix-points (smallest and greatest)
  It turns out it includes CTL*