A Compositional Approach to CTL* Verification

Amir Pnueli

Weizmann Institute of Sciences and New York University

CMU, August, 2003

Joint work with:

Yonit Kesten, Elad Shahar

Temporal Logics

Temporal Logic is widely used for the specification and verification of reactive systems and hardware designs.

There are (at least) two brands:

- LTL – Linear time logic, using the operators $\Box$, $\Diamond$, $\mathcal{U}$, and interpreted over individual computations.
- CTL – Branching time logic, using the operators $\mathbf{A}\Diamond$, $\mathbf{E}\Box$, and interpreted over computation (Kripke's) structures.

The Ever-Lasting Controversy

Since their introduction, there has been a continuous controversy about the relative merits of these two different brands of TLs.

Following are some of the arguments raised by the proponents of each camp:

Feature	CTL	LTL
Expressiveness Capabilities	$\mathbf{E}\Diamond p$	$\Box\Diamond p \rightarrow \Box\Diamond q$
Complexity of Model Checking	Linear	PSPACE-complete
Main method of Verification	Model Checking	Deductive

The Ever-Lasting Controversy

Since their introduction, there has been a continuous controversy about the relative merits of these two different brands of TLs.

Following are some of the arguments raised by the proponents of each camp:

Feature	CTL	LTL
Expressiveness	$\mathbf{E}\Diamond\, p$	$\Box\Diamond\, p \rightarrow \Box\Diamond\, q$
Complexity of Model Checking	Linear	PSPACE-complete
Main method of Verification	Model Checking	Deductive
Compositionality	Yes	No

Demonstrate that CTL is Compositional

Model checking CTL formulas can be viewed as a successive process of statification – finding for each temporal formula φ an assertion $\|\varphi\|$ which characterizes the set of all states satisfying φ.

Compositionality of CTL is illustrated by the equation

$$\|\mathbf{A}\Diamond\ \mathbf{A}\Box\ p\| = \|\mathbf{A}\Diamond\ (\|\mathbf{A}\Box\ p\|)\|,$$

stating that we can break the task of computing $\|\mathbf{A}\Diamond\ \mathbf{A}\Box\ p\|$ into the subtask of computing first an assertion $q = \|\mathbf{A}\Box\ p\|$ and then, computing $\|\mathbf{A}\Diamond\ q\|$.

Indeed, all model checking algorithms for CTL are incremental, dealing with one temporal operator at a time.

In contrast, model checking an LTL formula ψ traditionally starts with construction of a tableau which tackles the full formula ψ. No apparent compositionality or modularity there.

Main Message of this Talk

The main message of this talk is that CTL* (and therefore LTL) can be made compositional, but at a price.

Main Message of this Talk

The main message of this talk is that CTL* (and therefore LTL) can be made compositional, but at a price.

Obviously, there must be a price, otherwise we would have established

$$\text{PSPACE} = \text{P}.$$

Fair Discrete Systems

An FDS $\mathcal{D} = \langle V, \Theta, \rho, \mathcal{J}, \mathcal{C} \rangle$ consists of:

- V – A finite set of typed state variables. A V-state s is an interpretation of V. Σ_V – the set of all V-states.
- Θ – An initial condition. A satisfiable assertion that characterizes the initial states.
- ρ – A transition relation. An assertion $\rho(V, V')$, referring to both unprimed (current) and primed (next) versions of the state variables. For example, $x' = x + 1$ corresponds to the assignment $x := x + 1$.
- $\mathcal{J} = \{J_1, \ldots, J_k\}$ A set of justice (weak fairness) requirements. Ensure that a computation has infinitely many J_i-states for each J_i, $i = 1, \ldots, k$.
- $\mathcal{C} = \{\langle p_1, q_1 \rangle, \ldots \langle p_n, q_n \rangle\}$ A set of compassion (strong fairness) requirements. Infinitely many p_i-states imply infinitely many q_i-states.

An FDS provides a syntactic representation of fair Kripke structures. Note that every finite Kripke structure or one which is generated by a program or a circuit has a presentation as an FDS.

Computations

Let $\mathcal{D}$ be an FDS for which the above components have been identified. The state s' is defined to be a $\mathcal{D}$-successor of state s if

$$\langle s, s' \rangle \models \rho_{\mathcal{D}}(V, V').$$

We define a computation of $\mathcal{D}$ to be an infinite sequence of states

$$\sigma : s_0, s_1, s_2, ...,$$

satisfying the following requirements:

- Initiality: s_0 is initial, i.e., $s_0 \models \Theta$.
- Consecution: For each $j \geq 0$, the state s_{j+1} is a $\mathcal{D}$-successor of the state s_j.
- Justice: For each $J \in \mathcal{J}$, σ contains infinitely many J-positions
- Compassion: For each $\langle p, q \rangle \in \mathcal{C}$, if σ contains infinitely many p-positions, it must also contain infinitely many q-positions.

Synchronous Parallel Composition

The synchronous parallel composition of systems $\mathcal{D}_1$ and $\mathcal{D}_2$, denoted by $\mathcal{D}_1 \parallel\!| \mathcal{D}_2$, is given by the FDS $\mathcal{D} = \langle V, \Theta, \rho, \mathcal{J}, \mathcal{C} \rangle$, where

$$
\begin{array}{lcccc}
V & = & V_1 & \cup & V_2 \\
\Theta & = & \Theta_1 & \wedge & \Theta_2 \\
\rho & = & \rho_1 & \wedge & \rho_2 \\
\mathcal{J} & = & \mathcal{J}_1 & \cup & \mathcal{J}_2 \\
\mathcal{C} & = & \mathcal{C}_1 & \cup & \mathcal{C}_2
\end{array}
$$

Synchronous parallel composition is used for the construction of an observer: a system O which observes and evaluates the behavior of an observed system $\mathcal{D}$. Running $\mathcal{D} \parallel\!| O$, we let $\mathcal{D}$ behave as usual, while O observes its behavior.

A Unified Requirement Specification Language: the Temporal Logic CTL*

Assume an underlying (first-order) assertion language $\mathcal{L}$. The predicate at_ℓ_i, abbreviates the formula $\pi_j = \ell_i$, where ℓ_i is a location within process P_j.

A temporal formula is constructed out of assertions to which we apply the

- Boolean operators $\neg$, $\vee$, and $\wedge$,
- Temporal operators:

$\bigcirc$	– Next	$\mathcal{U}$	– Until	$\mathcal{W}$	– Waiting-for, Unless
$\ominus$	– Previous	$\mathcal{S}$	– Since	$\mathcal{B}$	– Back-to,

- Path quantifiers: **E**, **A**, $\mathbf{E}_f$, and $\mathbf{A}_f$.

Derived Temporal Operators

Additional temporal operators can be defined in terms of the basic ones as follows:

$$
\begin{array}{rcll}
\Diamond p & = & 1\,\mathcal{U}\,p & - \text{ Eventually} \\
\Box p & = & p\,\mathcal{W}\,0 & - \text{ Henceforth} \\
\diamondminus p & = & 1\,\mathcal{S}\,p & - \text{ Sometimes in the past} \\
\boxminus p & = & p\,\mathcal{B}\,0 & - \text{ Always in the past}
\end{array}
$$

CTL*: Syntax (1/2)

There are two types of sub-formulas in CTL*:

State formulas (interpreted over states):

- Every assertion in $\mathcal{L}$ is a state formula.
- If p is a path formula, then **E**p, **A**p, $\mathbf{E}_f p$ and $\mathbf{A}_f p$ are state formulas.
- If p and q are state formulas then so are $\neg p$, $p \vee q$, and $p \wedge q$.

Examples: p and **A**$\square\,(p \rightarrow \diamondsuit\, q)$ are state formulas.

CTL*: Syntax (2/2)

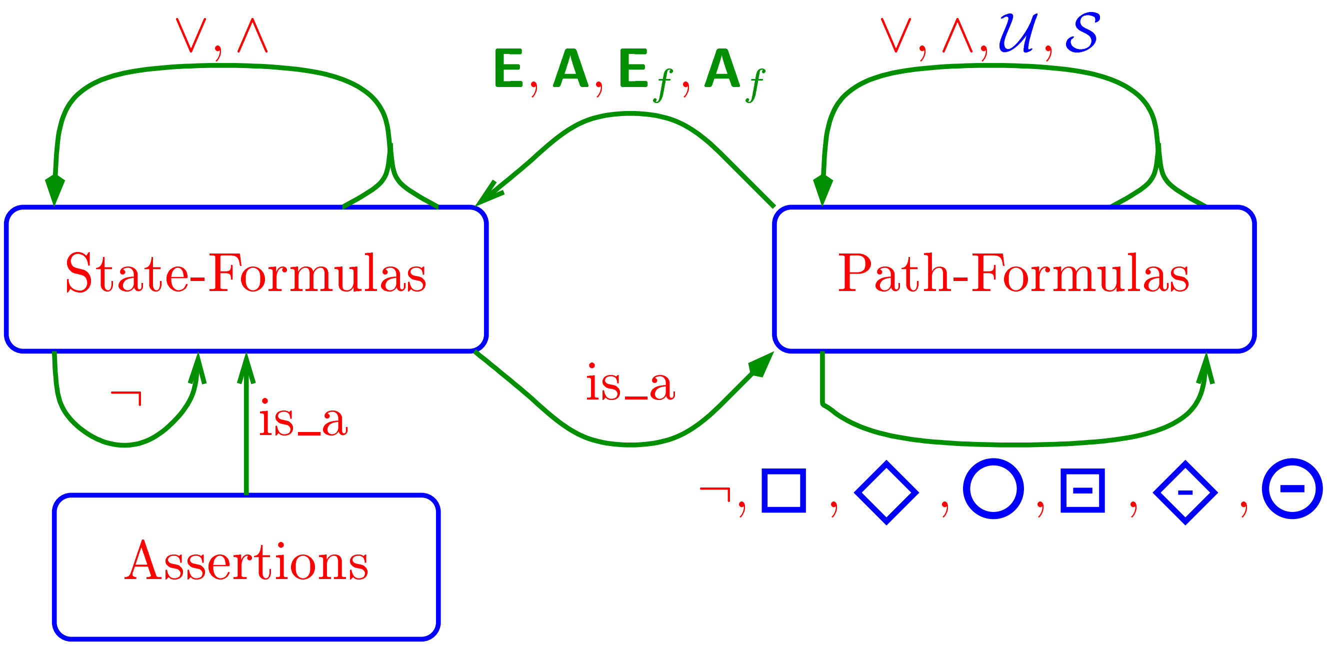
Path formulas (interpreted over state sequences):

- Every state formula is a path formula.
- If p and q are path formulas then so are $\neg p$, $p \vee q$, $p \wedge q$, $\bigcirc p$, $p\ \mathcal{U}\ q$, $p\ \mathcal{W}\ q$, $\ominus p$, $p\ \mathcal{S}\ q$, and $p\ \mathcal{B}\ q$

Examples: p and $\Box\Diamond \mathbf{E}\Diamond\ r$ are path formulas.

Any state formula is a CTL* formula. Path formulas which contain no path quantifiers are sometimes referred to as LTL formulas.

In Pictures


$\vee, \wedge$
$\mathbf{E}, \mathbf{A}, \mathbf{E}_f, \mathbf{A}_f$
$\vee, \wedge, \mathcal{U}, \mathcal{S}$
State-Formulas
Path-Formulas
$\neg$
is_a
is_a
$\neg, \square, \diamond, \bigcirc, \boxminus, \diamondminus, \ominus$
Assertions

Runs, Reachable, and Feasible States

Let $\mathcal{D}$ be an FDS. A run of $\mathcal{D}$ is an infinite sequence of states $\sigma : s_0, s_1, ...,$ satisfying the requirements of initiation and consecution, i.e., $s_0 \models \Theta$ and, for every $j \geq 0$, s_{j+1} is a $\mathcal{D}$-successor of s_j. We denote by $runs(\mathcal{D})$ the set of runs of $\mathcal{D}$.

Recall that a computation of $\mathcal{D}$ is a run which satisfies the requirements of justice and compassion.

A state s is said to be reachable if it participates in some run of $\mathcal{D}$. State s is feasible if it participates in some computation of $\mathcal{D}$.

CTL^*: Semantics

We interpret CTL^* formulas over (the computation structure of) an FDS $\mathcal{D}$. In the following, we use the term path as synonymous to a run of an FDS. Let $\pi : s_0, s_1, \ldots$ be a run of $\mathcal{D}$. Then, for $j \geq 0$, we write $\pi[j]$ to denote s_j, the jth state in π.

The semantics of CTL^* formulas is defined inductively as follows:

Interpretation of State Formulas

State formulas are interpreted over states in $\mathcal{D}$. We define the notion of a state formula p holding at a state s in $\mathcal{D}$, denoted $(\mathcal{D}, s) \models p$, as follows:

- For an assertion p,
  $(\mathcal{D}, s) \models p \quad \Longleftrightarrow \quad s \models p$
- For state formulas p and q,
  $(\mathcal{D}, s) \models \neg p \quad \Longleftrightarrow \quad$ It is not the case that $(\mathcal{D}, s) \models p$
  $(\mathcal{D}, s) \models p \vee q \quad \Longleftrightarrow \quad (\mathcal{D}, s) \models p$ or $(\mathcal{D}, s) \models q$
  $(\mathcal{D}, s) \models p \wedge q \quad \Longleftrightarrow \quad (\mathcal{D}, s) \models p$ and $(\mathcal{D}, s) \models q$
- For a path formula φ,

$(\mathcal{D}, s) \models \mathbf{E}\varphi \quad \Longleftrightarrow \quad (\mathcal{D}, \pi, j) \models \varphi$ for some path $\pi \in runs(\mathcal{D})$ and position $j \geq 0$ satisfying $\pi[j] = s$.

$(\mathcal{D}, s) \models \mathbf{A}\varphi \quad \Longleftrightarrow \quad (\mathcal{D}, \pi, j) \models \varphi$ for all paths $\pi \in runs(\mathcal{D})$ and positions $j \geq 0$ satisfying $\pi[j] = s$.

The semantics of $\mathbf{E}_f\varphi$ and $\mathbf{A}_f\varphi$ are defined similarly to $\mathbf{E}\varphi$ and $\mathbf{A}\varphi$ respectively, replacing path (run) by computation.

Interpretation of Path Formulas (1/2)

Path formulas are interpreted over runs of $\mathcal{D}$. We define the notion of a path formula p holding at a run $\pi \in runs(\mathcal{D})$ at position $j \geq 0$, denoted $(\mathcal{D}, \pi, j) \models p$, as follows:

- For a state formula p,

$(\mathcal{D}, \pi, j) \models p \quad \Longleftrightarrow \quad (\mathcal{D}, \pi[j]) \models p.$

- For path formulas p and q,

$(\mathcal{D}, \pi, j) \models \neg p \quad \Longleftrightarrow \quad$ It is not the case that $(\mathcal{D}, \pi, j) \models p$

$(\mathcal{D}, \pi, j) \models p \vee q \quad \Longleftrightarrow \quad (\mathcal{D}, \pi, j) \models p$ or $(\mathcal{D}, \pi, j) \models q$

$(\mathcal{D}, \pi, j) \models p \wedge q \quad \Longleftrightarrow \quad (\mathcal{D}, \pi, j) \models p$ and $(\mathcal{D}, \pi, j) \models q$

$(\mathcal{D}, \pi, j) \models \bigcirc p \quad \Longleftrightarrow \quad (\mathcal{D}, \pi, j+1) \models p$

$(\mathcal{D}, \pi, j) \models p \, \mathcal{U} \, q \quad \Longleftrightarrow \quad (\mathcal{D}, \pi, k) \models q$ for some $k \geq j$, and $(\mathcal{D}, \pi, i) \models p$ for all i, $j \leq i < k$

$(\mathcal{D}, \pi, j) \models p \, \mathcal{W} \, q \quad \Longleftrightarrow \quad (\mathcal{D}, \pi, j) \models p \, \mathcal{U} \, q$, or $(\mathcal{D}, \pi, i) \models p$ for all $i \geq j$

Interpretation of Path Formulas (2/2)

- For path formulas p and q,

$(\mathcal{D}, \pi, j) \models \ominus p \iff j > 0$ and $(\mathcal{D}, \pi, j-1) \models p$

$(\mathcal{D}, \pi, j) \models \circleddash\!\!\sim p \iff j = 0$ or $(\mathcal{D}, \pi, j-1) \models p$

$(\mathcal{D}, \pi, j) \models p\,\mathcal{S}\,q \iff (\mathcal{D}, \pi, k) \models q$ for some $k \leq j$, and $(\mathcal{D}, \pi, i) \models p$ for all i, $k < i \leq j$

$(\mathcal{D}, \pi, j) \models p\,\mathcal{B}\,q \iff (\mathcal{D}, \pi, j) \models p\,\mathcal{S}\,q$, or $(\mathcal{D}, \pi, i) \models p$ for all i, $0 \leq i \leq j$

Let φ be a CTL* formula. We say that φ holds on $\mathcal{D}$ (φ is $\mathcal{D}$-valid), denoted $\mathcal{D} \models \varphi$, if $(\mathcal{D}, s) \models \varphi$, for every initial state s in $\mathcal{D}$. A CTL* formula φ is called satisfiable if it holds on some model. A CTL* formula is called valid if it holds on all models.

Let p and q be CTL* formulas. We introduce the abbreviation

$p \Rightarrow q$ for $\mathbf{A}\square\,(p \rightarrow q)$.

where $p \rightarrow q$ is the logical implication equivalent to $\neg p \lor q$. Thus, the formula $p \Rightarrow q$ holds at $\mathcal{D}$ if the implication $p \rightarrow q$ holds at all reachable states.

Fragments of CTL*

- CTL – Path quantifiers and temporal operators always appear in the combination $\mathcal{QT}$, where $\mathcal{Q}$ is a path quantifier and $\mathcal{T}$ is a temporal operator.
- LTL – Formulas of the form $\mathbf{A}\varphi$, where φ is a path formula.
- ACTL – CTL* formulas where the only path quantifiers used are $\mathbf{A}$ and $\mathbf{A}_f$.

Temporal Testers

For every LTL formula φ, there exists an FDS T_φ called the temporal tester for φ. This tester has a distinguished boolean variable x, such that, in every σ, a computation of T_φ and every position $j \geq 0$, $x[s_j] = 1$ iff $(\sigma, j) \models \varphi$.

A path formula whose principal operator is temporal, and such that it does not contain any nested temporal operator or path quantifier is called a basic path formula.

We will only present testers for basic path formulas.

Example: a Tester for $\Diamond\, p$

$$T(\Diamond\, p) : \begin{cases} V: & \mathit{Vars}(p) \ \cup \ \{x\} \\ \Theta: & 1 \\ \rho: & x \quad = \quad p \ \vee \ x' \\ \mathcal{J}: & p \ \vee \ \neg x \\ \mathcal{C}: & \emptyset \end{cases}$$

The justice requirement demands that either $p = 1$ infinitely many times, or $x = 0$ infinitely many times. This rules out a computation in which $p = 0$ and $x = 1$ continuously, even though such a state sequence satisfies the requirements of initiality and consecution.

Testers for $\bigcirc p$ and $p\,\mathcal{U}\,q$

$$T(\bigcirc p) : \begin{cases} V : & \mathit{Vars}(p) \cup \{x\} \\ \Theta : & 1 \\ \rho : & x = p' \\ \mathcal{J} = \mathcal{C} : & \emptyset \end{cases}$$

$$T(p\,\mathcal{U}\,q) : \begin{cases} V : & \mathit{Vars}(p, q) \cup \{x\} \\ \Theta : & 1 \\ \rho : & x = q \vee (p \wedge x') \\ \mathcal{J} : & q \vee \neg x \\ \mathcal{C} : & \emptyset \end{cases}$$

Note the justice requirement by which either q or $x = 0$ should hold infinitely many times.

Model Checking CTL* Formulas

A state formula whose principal operators are a pair $\mathcal{Q}\mathcal{T}$ and which does not contain any additional temporal operators or path quantifiers is called a basic CTL formula ($|\mathcal{Q}| = |\mathcal{T}| = 1$).

A path formula whose principal operator is temporal, and such that it does not contain any nested temporal operators or path quantifiers is called a basic path formula ($|\mathcal{Q}| = 0, |\mathcal{T}| = 1$).

A basic state formula is a formula of the form $\mathcal{Q}\varphi$, where φ contains no path quantifiers ($|\mathcal{Q}| = 1$).

Statification

For a state formula φ, we denote by $\|\varphi\|$ the assertion characterizing the set of all $\mathcal{D}$-states satisfying φ. For the case that φ is an assertion, $\|\varphi\| = \varphi$.

Claim 1. [Model Checking State Formulas]
For a state formula φ,

$$\mathcal{D} \models \varphi \quad \textit{iff} \quad \Theta \to \|\varphi\|.$$

Thus, the essence of model checking is the computation of $\|\varphi\|$ for the various state formulas. We will provide a recipe for effective computation of the statification of all state formulas.

To shorten the presentation, we assume that we already know how to compute $\|\varphi\|$ for all basic CTL formulas φ. This is what every model checker does.

Model Checking General Future CTL Formulas

Let $f(\varphi)$ be a state formula containing one or more occurrences of the nested state formula φ, and let $q = \|\varphi\|$.

Claim 2. [Elimination of nested state formulas]

$$\|f(\varphi)\| \quad = \quad \|f(q)\|,$$

where $f(q)$ is obtained by substituting q for all occurrences of φ in f.

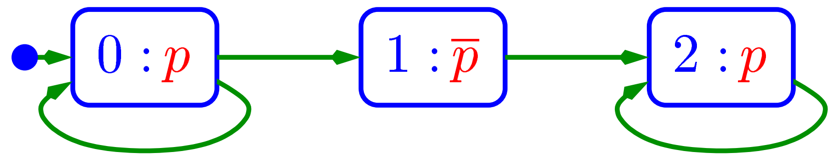
Consider a general future CTL formula. Claim 2 enables us to eliminate all nested state formulas, starting with the innermost ones, successively applying the known techniques for statification of basic CTL formulas.

The statement of Claim 2 can also be phrased as:

$$\|f(\varphi)\| \quad = \quad \|f(\|\varphi\|)\|.$$

Example

$\mathcal{D}$:



Try to model check the formula $f = \mathbf{A}\Diamond \underbrace{\mathbf{A}\Box\, p}_{\varphi}$. Following Claim 2, we compute first

$\|\varphi\| = \|\mathbf{A}\Box\, p\| = (\pi = 2).$

Next, we compute

$\|f(\|\varphi\|)\| = \|\mathbf{A}\Diamond\, (\pi = 2)\| = (\pi > 0).$

Now, it remains to check

$\Theta \rightarrow \|f\| \quad = \quad (\pi = 0 \wedge p \ \rightarrow \ \pi > 0) \quad = \quad 0,$

which shows that $\mathbf{A}\Diamond\mathbf{A}\Box\, p$ does not hold on $\mathcal{D}$.

Elimination of Temporal Operators

The modularity of CTL which enabled us to model check a formula by successively computing $\|\varphi\|$ for a sequence of nested basic CTL formulas has, for a long time, been considered a unique feature of CTL, and a major argument in the branching vs. linear battle.

A similar modularity (though for a higher price) exists for the LTL component of a general CTL* formula, as shown by the following:

Claim 3. [Elimination of Temporal Operators]
Let $f(\psi)$ be a basic state formula containing one or more occurrences of the basic path formula ψ. Then, we can compute

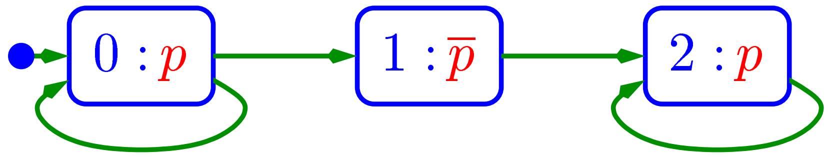
$$\|f(\psi)\|_{\mathcal{D}} \quad = \quad \Big(\|f(x)\|_{\mathcal{D}|||T_\psi}\Big)\Downarrow_V,$$

where T_ψ is the temporal tester for ψ, x is the fresh variable introduced by T_ψ, and $\Downarrow_V$ is a projection operator which removes from an assertion (by existential quantification) all the variables not in V. The expression $\|f(x)\|_{\mathcal{D}|||T_\psi}$ stands for the statification of $f(x)$ computed over the augmented FDS $\mathcal{D} \mathrel{|||} T_\psi$.

Example 1/2

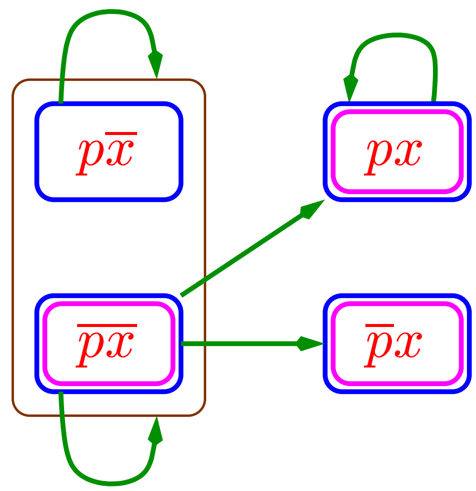
Consider the system

$\mathcal{D}:$



We wish to model check the property $\mathbf{A}_f \Diamond \Box\, p$. First, we construct the tester $T_{\Box p}$.

$$T_{\Box p}: \left\{ \begin{array}{rl} V = \mathcal{O}: & \{p, x\} \\ \Theta: & 1 \\ \rho: & x \;=\; p \;\wedge\; x' \\ \mathcal{J}: & x \;\vee\; \neg p \\ \mathcal{C}: & \emptyset \end{array} \right.$$

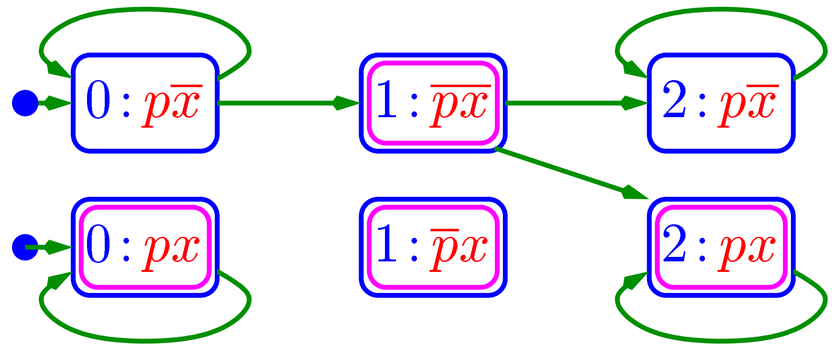


The justice requirement $x \vee \neg p$ is intended to guarantee that we will not have a computation in which continuously $p = 1$, while $x = 0$.

Example 2/2

Next, we form the parallel composition $\mathcal{D}_+ : \mathcal{D} \parallel\!| \; T_{\Box p}$.

$\mathcal{D}_+$:



Evaluating $\|\mathbf{A}_f \Diamond x\|$ over $\mathcal{D}_+$, we obtain $\|\mathbf{A}_f \Diamond x\| = 1$. We can therefore conclude that the original FDS $\mathcal{D}$ satisfies $\mathbf{A}_f \Diamond \Box p$.

Deductive Verification

Based on theorem proving techniques, the method of deductive verification can be used to establish temporal properties of infinite-state reactive systems.

We assume that all CTL* formulas are given in a positive normal form, i.e., negations are only applied to assertions.

For simplicity of the presentation, we consider systems with no compassion requirements.

Structure of the Proof System

The structure of the deductive system we present is as follows:

- Rules for each of the basic CTL formulas, i.e. formulas of the form $\mathcal{Q}\mathcal{T}p$ where p is an assertion.
- A reduction rule which enables us to decompose the verification task into several subtasks, each dealing with a single basic state formula. Recall that a basic state formula is a formula of the form $\mathcal{Q}\varphi$, where φ contains no path quantifiers.
- A reduction rule which enables us to eliminate one basic path formula at a time, at the cost of conjoining a tester for that formula to the system we are verifying. Recall that a basic path formula is a formula of the form $\mathcal{T}p$, where $\mathcal{T}$ is a temporal operator and p is an assertion.

Preliminary Rules

We assume the availability of an underlying proof system for assertional reasoning. Following is the Generalization rule,

For an assertion p,

$$\frac{\vdash_{FO} \quad p}{\vdash \quad \mathbf{A}\Box\, p}$$

stating that an assertion that has been proved to be generally valid holds, in particular, on any reachable state.

The following Entailment Modus Ponens rule enables us to perform propositional reasoning uniformly at all reachable states. Recall that $p \Rightarrow q$ is an abbreviation for $\mathbf{A}\Box\,(p \rightarrow q)$.

For state formulas p and q,

$$\frac{\mathbf{A}\Box\, p, \quad p \Rightarrow q}{\mathbf{A}\Box\, q}$$

Universal Invariance

The following rule A-INV can be used to prove that if assertion p holds at state s, then q holds at all states reachable from s.

For assertions p, q, and φ,

$$\begin{array}{ll} \text{I1.} & p \Rightarrow \varphi \\ \text{I2.} & \varphi \Rightarrow q \\ \text{I3.} & \varphi \wedge \rho \Rightarrow \varphi' \\ \hline & p \Rightarrow \mathbf{A}\Box\, q \end{array}$$

The auxiliary assertion φ is often described as an inductive strengthening of q. The rule itself is based on computational induction.

Finding φ and similar auxiliary constructs is one of the most challenging problems in the application of deductive verification, and requires ingenuity and insight.

Example: MUX-SEM

y : **natural initially** $y = 1$

$$\left[\begin{array}{l} \ell_0: \textbf{loop forever do} \\ \left[\begin{array}{ll} \ell_1: & \textbf{Non-critical} \\ \ell_2: & \textbf{request } y \\ \ell_3: & \textbf{Critical} \\ \ell_4: & \textbf{release } y \end{array}\right] \end{array}\right] \quad \| \quad \left[\begin{array}{l} m_0: \textbf{loop forever do} \\ \left[\begin{array}{ll} m_1: & \textbf{Non-critical} \\ m_2: & \textbf{request } y \\ m_3: & \textbf{Critical} \\ m_4: & \textbf{release } y \end{array}\right] \end{array}\right]$$

Wishing to establish mutual exclusion, we use rule A-INV to prove

$$\Theta \quad \Rightarrow \quad \mathbf{A}\Box\, \neg(at\_\ell_3 \;\wedge\; at\_m_3)$$

As the inductive assertion φ we choose:

$$\neg(at\_\ell_{3,4} \;\wedge\; at\_m_{3,4}) \;\wedge\; (at\_\ell_{3,4} \rightarrow y = 0) \;\wedge\; (at\_m_{3,4} \rightarrow y = 0)$$

Note that the last two conjuncts form assertions attached at the locations ℓ_3, ℓ_4, m_3, m_4.

Rule E-NEXT

Following is rule E-NEXT:

For assertions p and q

$$\frac{\text{N1.} \quad p \;\Rightarrow\; \exists V' : \rho \;\wedge\; q'}{p \;\Rightarrow\; \mathbf{E}\bigcirc q}$$

This rule can be used to establish that every p-state has a successor satisfying q

Rule E-UNTIL

Following is rule E-UNTIL:

For assertions p, q, r, and φ
a well-founded domain $(\mathcal{A}, \succ)$,
and a ranking function $\delta : \Sigma \mapsto \mathcal{A}$

$$\begin{array}{ll} \text{U1.} & p \;\Rightarrow\; \varphi \\ \text{U2.} & \varphi \;\Rightarrow\; r \;\vee\; (q \;\wedge\; \exists V' : (\rho \;\wedge\; \varphi' \;\wedge\; \delta \succ \delta')) \\ \hline & p \;\Rightarrow\; q\, \mathbf{E}\mathcal{U}\, r \end{array}$$

The rule uses a well-founded domain $(\mathcal{A}, \succ)$, consisting of a set $\mathcal{A}$ and an order relation $\succ$, such that there does not exist an infinitely descending chain of $\mathcal{A}$-elements:

$$a_0 \succ a_1 \succ a_2 \succ \cdots$$

The rule also uses a ranking function δ mapping states of the system into the well founded domain $\mathcal{A}$.

Example: BAKERY-2

local y_1, y_2 : **natural initially** $y_1 = y_2 = 0$

$$\left[\begin{array}{l} \ell_0 : \textbf{loop forever do} \\ \left[\begin{array}{l} \ell_1 : \textbf{Non-Critical} \\ \ell_2 : y_1 := y_2 + 1 \\ \ell_3 : \textbf{await} \left(\begin{array}{ll} y_2 = 0 & \vee \\ y_1 < y_2 & \end{array}\right) \\ \ell_4 : \textbf{Critical} \\ \ell_5 : y_1 := 0 \end{array}\right] \end{array}\right] \quad \| \quad \left[\begin{array}{l} m_0 : \textbf{loop forever do} \\ \left[\begin{array}{l} m_1 : \textbf{Non-Critical} \\ m_2 : y_2 := y_1 + 1 \\ m_3 : \textbf{await} \left(\begin{array}{ll} y_1 = 0 & \vee \\ y_2 \leq y_1 & \end{array}\right) \\ m_4 : \textbf{Critical} \\ m_5 : y_2 := 0 \end{array}\right] \end{array}\right]$$

We prove, using rule E-UNTIL the property $\Theta \Rightarrow (1\, \mathbf{E}\mathcal{U}\ at\_\ell_4)$, claiming that it is possible for process P_1 to get to the critical section, starting at the initial state. We choose as follows:

$$\begin{array}{ll} p: & \Theta \\ q: & 1 \\ r: & at\_\ell_4 \\ \varphi: & at\_\ell_{0..4} \ \wedge \ at\_m_0 \ \wedge \ y_2 = 0 \\ (\mathcal{A}, \succ): & (\mathbb{N}, >) \\ \delta: & 4 - \textit{num}(\pi_1) \end{array}$$

where $\textit{num}(\pi_1)$ is the function which yields the natural j if $\pi_1 = \ell_j$.

Rule E-INV

For assertions p, $\varphi_0, \ldots, \varphi_m$,
an FDS whose justice requirements are $J_1, \ldots, J_m \in \mathcal{J}$,
and taking $J_0 = 1$.

$$\text{I1.} \quad p \Rightarrow \bigvee_{i=0}^{m} \varphi_i$$

For $i = 0, \ldots, m$,

$$\text{I2.} \quad \varphi_i \Rightarrow J_i$$

$$\text{I3.} \quad \varphi_i \Rightarrow q \wedge \mathbf{E}\bigcirc(q\,\mathbf{E}\mathcal{U}\,\varphi_{i \oplus_m 1})$$

$$\overline{\qquad p \Rightarrow \mathbf{E}_f\square\, q \qquad}$$

For $i < m$, $i \oplus_m 1 = i + 1$, while $m \oplus_m 1 = 0$. The rule requires identifying auxiliary assertions $\varphi_0, \ldots, \varphi_m$, such that each φ_i implies J_i, and there exists a computation which visits $\varphi_0, \ldots, \varphi_m$ in a round-robin fashion.

Universal Response

For justice requirements $J_1, \ldots, J_m$,
assertions $p, q, h_1, \ldots, h_m$,
well-founded domain $(\mathcal{A}, \succ)$,
and ranking functions $\delta_1, \ldots, \delta_m : \Sigma \mapsto \mathcal{A}$

W1. $p \Rightarrow q \vee \bigvee_{j=1}^{m} h_j$

W2. For $i = 1, \ldots, m$

$$h_i \wedge \rho \Rightarrow q' \vee (\neg J_i' \wedge h_i' \wedge \delta_i = \delta_i') \vee \left(\bigvee_{j=1}^{m} h_j' \wedge (\delta_i \succ \delta_j')\right)$$

$$p \Rightarrow \mathbf{A}_f \Diamond q$$

Example: BAKERY-2

local y_1, y_2 : **natural initially** $y_1 = y_2 = 0$

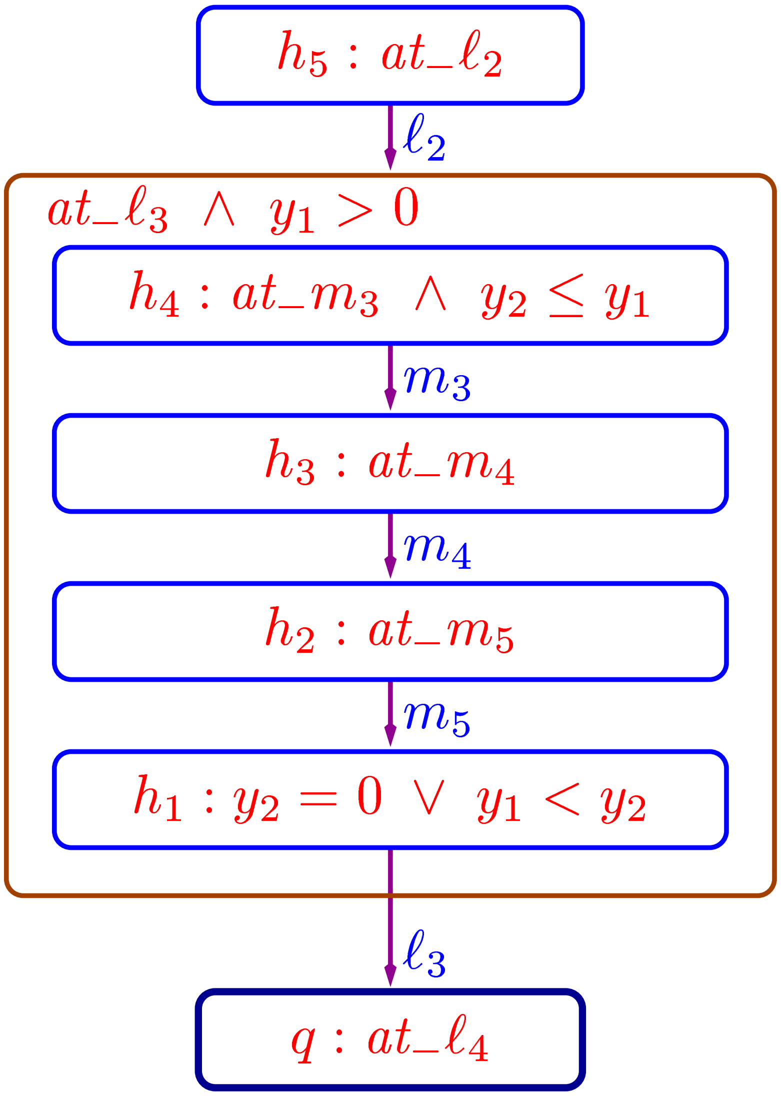
$$\left[\begin{array}{l} \ell_0 : \textbf{loop forever do} \\ \left[\begin{array}{l} \ell_1 : \textbf{Non-Critical} \\ \ell_2 : y_1 := y_2 + 1 \\ \ell_3 : \textbf{await} \begin{pmatrix} y_2 = 0 & \vee \\ y_1 < y_2 & \end{pmatrix} \\ \ell_4 : \textbf{Critical} \\ \ell_5 : y_1 := 0 \end{array}\right] \end{array}\right] \parallel \left[\begin{array}{l} m_0 : \textbf{loop forever do} \\ \left[\begin{array}{l} m_1 : \textbf{Non-Critical} \\ m_2 : y_2 := y_1 + 1 \\ m_3 : \textbf{await} \begin{pmatrix} y_1 = 0 & \vee \\ y_2 \leq y_1 & \end{pmatrix} \\ m_4 : \textbf{Critical} \\ m_5 : y_2 := 0 \end{array}\right] \end{array}\right]$$

We prove, using rule A-RESP the property $at\_\ell_2 \Rightarrow \mathbf{A}_f \Diamond\, at\_\ell_4$. We choose $p = at\_\ell_2$, $q = at\_\ell_4$, $(\mathcal{A}, \succ) = (\mathbb{N}, >)$, and

i	J_i	h_i	δ_i
1	$\neg(at\_\ell_3 \ \wedge \ (y_2 = 0 \ \vee \ y_1 < y_2))$	$\neg J_1$	1
2	$\neg at\_m_5$	$at\_\ell_3 \ \wedge \ at\_m_5$	2
3	$\neg at\_m_4$	$at\_\ell_3 \ \wedge \ at\_m_4$	3
4	$\neg(at\_m_3 \ \wedge \ (y_1 = 0 \ \vee \ y_2 \leq y_1))$	$at\_\ell_3 \ \wedge \ \neg J_4$	4
5	$\neg at\_\ell_2$	at_ℓ_2	5

Verification Diagrams

An A-RESP proof can also be presented in a verification diagram.



Decomposing a Proof into Proofs of Basic State Formulas

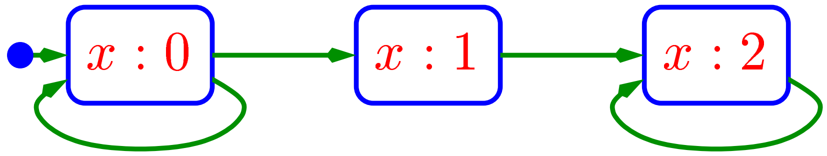
Recall that a basic state formula is a formula of the form $\mathcal{Q}\varphi$, where φ contains no path quantifiers.

The following rule BASIC-STATE allows us to decompose a proof of an arbitrary state formula into proofs of basic state formulas:

For a formula $f(\varphi)$, containing occurrences of the basic state formula φ, and an assertion p,

$$\begin{array}{ll} \text{R1.} & p \Rightarrow \varphi \\ \text{R2.} & f(p) \\ \hline & f(\varphi) \end{array}$$

Example



We wish to prove for this system the property f : $\mathbf{E}\Box\ \mathbf{E}\Diamond\ (x = 1)$, claiming the existence of a run from each of whose states it is possible to reach a state at which $x = 1$.

Using BASIC-STATE, it is possible to reduce the task of verifying the non-basic formula $\mathbf{E}\Box\ \mathbf{E}\Diamond\ (x = 1)$ into the two tasks of verifying

R1. $(x = 0) \Rightarrow \mathbf{E}\Diamond\ (x = 1)$
R2. $\mathbf{E}\Box\ (x = 0)$

Note that, as the assertion p, we have chosen $x = 0$. The design of an appropriate assertion p which characterizes states satisfying φ is the part which requires creativity and ingenuity in the application of BASIC-STATE.

Eliminating Basic Path Formulas

Recall that a basic path formula is a path formula whose principal operator is temporal and which does not contain any additional temporal operators or path quantifiers.

The following rule BASIC-PATH enables to eliminate basic path formulas from a bigger formula:

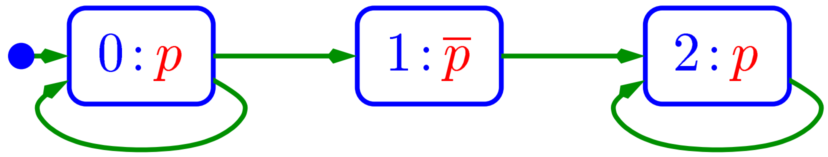
> For a fair basic state formula $f(\varphi)$, containing occurrences of the basic path formula φ, and an FDS $\mathcal{D}$,
>
> $$\frac{\mathcal{D} \;|||\; T_\varphi \quad \vdash \quad f(x_\varphi)}{\mathcal{D} \quad \vdash \quad f(\varphi)}$$

where x_φ is the fresh variable introduced by the tester $T\varphi$.

Example 1/2

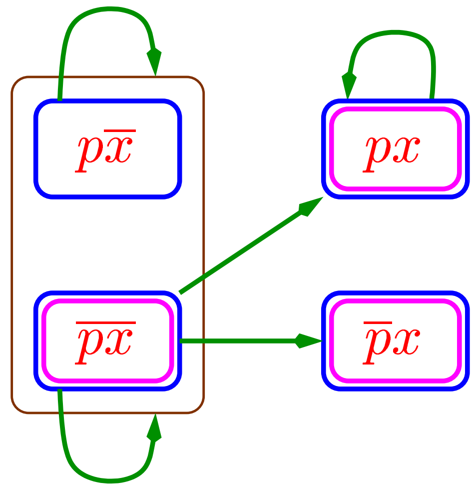
Consider the system

$\mathcal{D}$:



We wish to verify $\mathcal{D} \models \mathbf{A}_f \Diamond \Box\, p$. First, we construct the tester $T_{\Box p}$.

$$T_{\Box p} : \begin{cases} V : & \{p, x\} \\ \Theta : & 1 \\ \rho : & x \; = \; p \wedge x' \\ \mathcal{J} : & x \vee \neg p \\ \mathcal{C} : & \emptyset \end{cases}$$

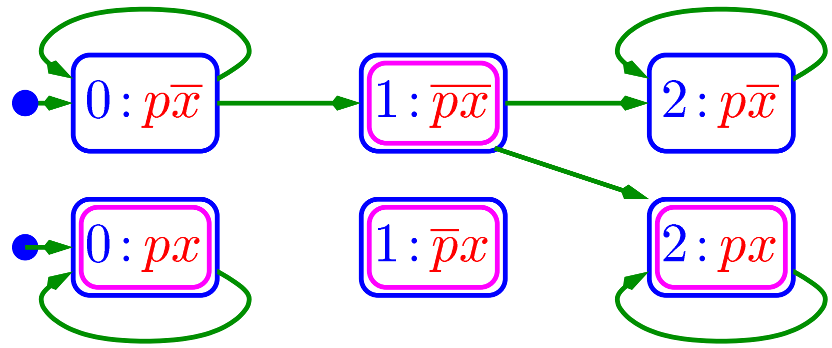


The justice requirement $x \vee \neg p$ is intended to guarantee that we will not have a computation in which continuously $p = 1$, while $x = 0$.

Example 2/2

Next, we form the parallel composition $\mathcal{D}_+ : \mathcal{D} \parallel\!| \; T_{\square p}$.

$\mathcal{D}_+$:



We need to verify $\mathcal{D}_+ \models \mathbf{A}_f \diamondsuit\, x$. The rule for Universal Response is adequate for proving $\mathcal{D}_+ \models (\pi = 0) \Rightarrow \mathbf{A}_f \diamondsuit\, x$. We can therefore conclude that the original FDS $\mathcal{D}$ satisfies $\mathbf{A}_f \diamondsuit \square\, p$.

Re-Completing the Temporal Picture

In a paper

[MP91] – Z. Manna and Pnueli, Completing the Temporal Picture.

and the two books with Zohar, we presented a complete proof theory for LTL. The theory included few select rules for the properties of invariance ($\square\, p$), response ($p \Rightarrow \diamondsuit\, q$), and reactivity ($\square \diamondsuit\, p \Rightarrow \square \diamondsuit\, q$). The claim for completeness was based on the presentation of every temporal formula in a canonic form which is a conjunction of reactivity properties, where p and q are arbitrary past formulas.

The results reported here present an even more complete picture, where we showed that the theory can be extended to full CTL* and completeness can be based on the two reduction principles which successively eliminate basic state formulas and basic path formulas.

Conclusions

- CTL* can be verified in a compositional manner, based on the following reduction principles:
  - Decomposing the proof into proofs of basic state formulas – getting rid of one path quantifier at a time.
  - Elimination of basic path formulas at the price of introducing a tester for the formula – getting rid of one temporal operator at a time.
- We presented a novel (relatively) complete deductive system for CTL*.
- Proposed a new (tester-based) and more effective answer to the old question "how to verify an arbitrary LTL formula?"
- Technically, the work presents a modular tableau construction.