Interpolation-Based Model Checking

Viktor Kuncak, EPFL

`https://lara.epfl.ch/w/fv`

Interpolants for Implication

Definition

Let F, G be propositional formulas. An **interpolant** for implication (F, G) is a formula H such that the following three conditions hold:

- $F \models H$
- $H \models G$
- $FV(H) \subseteq FV(F) \cap FV(G)$

Note that these conditions imply that $F \models G$. Formulas H is between F and G and serves as an *explanation* why F implies G, without referring to variables that are specific to F or specific to G.

Existence and Lattice of Interpolants

Theorem

Let F, G be propositional formulas such that $F \models G$ and let S be the set of interpolants (F, G). Then the following hold:

- *If $H_1, H_2 \in S$ then $H_1 \wedge H_2 \in S$ and $H_1 \vee H_2 \in S$*
- *S is non-empty*
- *There exists $H_{min} \in S$ such that, for all $H \in S$, $H_{min} \models H$*
- *There exists $H_{max} \in S$ such that, for all $H \in S$, $H \models H_{min}$*

Existence and Lattice of Interpolants

Theorem

Let F, G be propositional formulas such that $F \models G$ and let S be the set of interpolants (F, G). Then the following hold:

- *If $H_1, H_2 \in S$ then $H_1 \wedge H_2 \in S$ and $H_1 \vee H_2 \in S$*
- *S is non-empty*
- *There exists $H_{min} \in S$ such that, for all $H \in S$, $H_{min} \models H$*
- *There exists $H_{max} \in S$ such that, for all $H \in S$, $H \models H_{min}$*

Condition $F(\bar{x}, \bar{y}) \models I(\bar{y}) \models G(\bar{y}, \bar{z})$ can be written as the truth of the formulas

$$\forall \bar{x}, \bar{y}, \bar{z}.\ (F(\bar{x}, \bar{y}) \to I(\bar{y}) \wedge I(\bar{y}) \to G(\bar{y}, \bar{z})) \quad \text{i.e.}$$

$$\forall \bar{y}.\ (((\exists \bar{x}. F(\bar{x}, \bar{y})) \to I(\bar{y})) \wedge (I(\bar{y}) \to \forall \bar{z}. G(\bar{y}, \bar{z})))$$

Take as H_{min} and H_{max} results of expanding quantifiers in $\exists \bar{x}. H$, resp. $\forall \bar{z}. G$.

Interpolants for Unsat Conjunction - Reverse Interpolants

Instead of validity of $F \rightarrow G$, we look at unsatisfiability of $F \wedge \neg G$ (let B denote $\neg G$)
Let A, B be sets of clauses such that $A \wedge B$ is not satisfiable. (From now on) an interpolant for $A \wedge B$ is a formula H such that

- $A \models H$, $B \models \neg H$, and
- $FV(H) \subseteq FV(A) \cap FV(B)$

If $A \wedge B$ is unsatisfiable, if we transform it to conjunctive form, we can derive empty clause using resolution.

Interpolants for Unsat Conjunction - Reverse Interpolants

Instead of validity of $F \rightarrow G$, we look at unsatisfiability of $F \wedge \neg G$ (let B denote $\neg G$)
Let A, B be sets of clauses such that $A \wedge B$ is not satisfiable. (From now on) an interpolant for $A \wedge B$ is a formula H such that

- $A \models H$, $B \models \neg H$, and
- $FV(H) \subseteq FV(A) \cap FV(B)$

If $A \wedge B$ is unsatisfiable, if we transform it to conjunctive form, we can derive empty clause using resolution.

Can we use this resolution proof to construct an interpolant for $A \wedge B$?

Interpolants for Unsat Conjunction - Reverse Interpolants

Instead of validity of $F \to G$, we look at unsatisfiability of $F \wedge \neg G$ (let B denote $\neg G$)
Let A, B be sets of clauses such that $A \wedge B$ is not satisfiable. (From now on) an interpolant for $A \wedge B$ is a formula H such that

- $A \models H$, $B \models \neg H$, and
- $FV(H) \subseteq FV(A) \cap FV(B)$

If $A \wedge B$ is unsatisfiable, if we transform it to conjunctive form, we can derive empty clause using resolution.

Can we use this resolution proof to construct an interpolant for $A \wedge B$?
Yes!

Tseytin's Transformation and Interpolants

An interpolant for $A \wedge B$ is a formula H such that

- $A \models H$, $B \models \neg H$, and
- $FV(H) \subseteq FV(A) \cap FV(B)$

Tseytin's transformations $cnf(A)$, $cnf(B)$ introduce fresh variables $\bar{p}_A, \bar{p}_B$ where

$$\models A \longleftrightarrow \exists \bar{p}_A . cnf(A) \quad \text{and} \quad \models B \longleftrightarrow \exists \bar{p}_B . cnf(B)$$

Is interpolant H for $(cnf(A), cnf(B))$ also an interpolant for A and B?

Tseytin's Transformation and Interpolants

An interpolant for $A \wedge B$ is a formula H such that

- $A \models H$, $B \models \neg H$, and
- $FV(H) \subseteq FV(A) \cap FV(B)$

Tseytin's transformations $cnf(A)$, $cnf(B)$ introduce fresh variables $\bar{p}_A, \bar{p}_B$ where

$$\models A \longleftrightarrow \exists \bar{p}_A . cnf(A) \quad \text{and} \quad \models B \longleftrightarrow \exists \bar{p}_B . cnf(B)$$

Is interpolant H for $(cnf(A), cnf(B))$ also an interpolant for A and B?

$\models \forall \bar{p}_A . (cnf(A) \rightarrow H)$

Tseytin's Transformation and Interpolants

An interpolant for $A \wedge B$ is a formula H such that

- $A \models H$, $B \models \neg H$, and
- $FV(H) \subseteq FV(A) \cap FV(B)$

Tseytin's transformations $cnf(A)$, $cnf(B)$ introduce fresh variables $\bar{p}_A, \bar{p}_B$ where

$$\models A \longleftrightarrow \exists \bar{p}_A . cnf(A) \quad \text{and} \quad \models B \longleftrightarrow \exists \bar{p}_B . cnf(B)$$

Is interpolant H for $(cnf(A), cnf(B))$ also an interpolant for A and B?

$\models \forall \bar{p}_A . (cnf(A) \rightarrow H)$ so $\models (\exists \bar{p}_A . cnf(A)) \rightarrow H$ and thus $\models A \rightarrow H$ i.e. $A \models H$
$\models \forall \bar{p}_B . (cnf(B) \rightarrow \neg H)$

Tseytin's Transformation and Interpolants

An interpolant for $A \wedge B$ is a formula H such that

- $A \models H$, $B \models \neg H$, and
- $FV(H) \subseteq FV(A) \cap FV(B)$

Tseytin's transformations $cnf(A)$, $cnf(B)$ introduce fresh variables $\bar{p}_A, \bar{p}_B$ where

$$\models A \longleftrightarrow \exists \bar{p}_A . cnf(A) \quad \text{and} \quad \models B \longleftrightarrow \exists \bar{p}_B . cnf(B)$$

Is interpolant H for $(cnf(A), cnf(B))$ also an interpolant for A and B?

$\models \forall \bar{p}_A . (cnf(A) \rightarrow H)$ so $\models (\exists \bar{p}_A . cnf(A)) \rightarrow H$ and thus $\models A \rightarrow H$ i.e. $A \models H$
$\models \forall \bar{p}_B . (cnf(B) \rightarrow \neg H)$ so $\models (\exists \bar{p}_B . cnf(B)) \rightarrow \neg H$ and thus $\models B \rightarrow \neg H$ i.e. $B \models \neg H$
$FV(H) \subseteq FV(cnf(A)) \cap FV(cnf(B))$

Tseytin's Transformation and Interpolants

An interpolant for $A \wedge B$ is a formula H such that

- $A \models H$, $B \models \neg H$, and
- $FV(H) \subseteq FV(A) \cap FV(B)$

Tseytin's transformations $cnf(A)$, $cnf(B)$ introduce fresh variables $\bar{p}_A, \bar{p}_B$ where

$$\models A \longleftrightarrow \exists \bar{p}_A . cnf(A) \quad \text{and} \quad \models B \longleftrightarrow \exists \bar{p}_B . cnf(B)$$

Is interpolant H for $(cnf(A), cnf(B))$ also an interpolant for A and B?

$\models \forall \bar{p}_A . (cnf(A) \rightarrow H)$ so $\models (\exists \bar{p}_A . cnf(A)) \rightarrow H$ and thus $\models A \rightarrow H$ i.e. $A \models H$
$\models \forall \bar{p}_B . (cnf(B) \rightarrow \neg H)$ so $\models (\exists \bar{p}_B . cnf(B)) \rightarrow \neg H$ and thus $\models B \rightarrow \neg H$ i.e. $B \models \neg H$
$FV(H) \subseteq FV(cnf(A)) \cap FV(cnf(B)) \subseteq (FV(A) \cup \bar{p}_A) \cap (FV(B) \cup \bar{p}_B)$

Tseytin's Transformation and Interpolants

An interpolant for $A \wedge B$ is a formula H such that

- $A \models H$, $B \models \neg H$, and
- $FV(H) \subseteq FV(A) \cap FV(B)$

Tseytin's transformations $cnf(A)$, $cnf(B)$ introduce fresh variables $\bar{p}_A, \bar{p}_B$ where

$$\models A \longleftrightarrow \exists \bar{p}_A . cnf(A) \quad \text{and} \quad \models B \longleftrightarrow \exists \bar{p}_B . cnf(B)$$

Is interpolant H for $(cnf(A), cnf(B))$ also an interpolant for A and B?

$\models \forall \bar{p}_A . (cnf(A) \rightarrow H)$ so $\models (\exists \bar{p}_A . cnf(A)) \rightarrow H$ and thus $\models A \rightarrow H$ i.e. $A \models H$
$\models \forall \bar{p}_B . (cnf(B) \rightarrow \neg H)$ so $\models (\exists \bar{p}_B . cnf(B)) \rightarrow \neg H$ and thus $\models B \rightarrow \neg H$ i.e. $B \models \neg H$
$FV(H) \subseteq FV(cnf(A)) \cap FV(cnf(B)) \subseteq (FV(A) \cup \bar{p}_A) \cap (FV(B) \cup \bar{p}_B) = FV(A) \cap FV(B)$

Interpolants from Resolution Proofs

Assume we work with clauses. Construct resolution tree that derives a contradiction from $A \cup B$. The tree has elements of $A \cup B$ as leaves and results of application of resolution as nodes. For each clause C in resolution tree we define recursively formula $I(C)$ and show that $I(C)$ is interpolant for these two formulas:

- $A \wedge (\neg C|_A)$
- $B \wedge (\neg C|_B)$

where $C|_A$ denotes the clause with only those literals from C whose propositional variables belong to $FV(A)$.
It follows that for empty clause the $I(\emptyset)$ is the interpolant for (A, B).

Interpolant Strength, by D'Silva, Kröning, Purandare, Weissenbacher, 2009

Construction of $I(C)$: Symmetric System

One construction of $I(C)$:

- if $C \in A \cup B$ is source in the proof DAG, then:
  - if $C \in A$ then $I(C) = 0$
  - if $C \in B$ then $I(C) = 1$
- if C is result of applying resolution to C_1, C_2 along propositional variable q where $\neg q \in C_1$, and $q \in C_2$, then
  - if $q \in FV(A) \setminus FV(B)$ then $I(C) = I(C_1) \vee I(C_2)$
  - if $q \in FV(B) \setminus FV(A)$ then $I(C) = I(C_1) \wedge I(C_2)$
  - if $q \in FV(A) \cap FV(B)$ then $I(C) = q\,?\,I(C_1) : I(C_2)$

We prove that $I(C)$ has the desired property by induction on the structure of the resolution proof tree.

Another Construction of $I(C)$: McMillan's System

K.L. McMillan. Interpolation and SAT-Based Model Checking. CAV 2013.

Define global $gl(C) = C|_{FV(A)\cap FV(B)}$ and local $loc(C) = C|_{FV(A)\setminus FV(B)}$

- if $C \in A \cup B$ is source in the proof DAG then
  - if $C \in A$ then $I(C) = gl(C)$
  - else $I(C) = 1$
- if C is result of applying resolution to C_1, C_2 along propositional variable q where $\neg q \in C_1$, and $q \in C_2$, then
  - if $q \in FV(A) \setminus FV(B)$ then $I(C) = I(C_1) \vee I(C_2)$
  - else $I(C) = I(C_1) \wedge I(C_2)$

Another Construction of $I(C)$: McMillan's System

K.L. McMillan. Interpolation and SAT-Based Model Checking. CAV 2013.

Define global $gl(C) = C|_{FV(A)\cap FV(B)}$ and local $loc(C) = C|_{FV(A)\setminus FV(B)}$

- if $C \in A \cup B$ is source in the proof DAG then
  - if $C \in A$ then $I(C) = gl(C)$
  - else $I(C) = 1$
- if C is result of applying resolution to C_1, C_2 along propositional variable q where $\neg q \in C_1$, and $q \in C_2$, then
  - if $q \in FV(A) \setminus FV(B)$ then $I(C) = I(C_1) \vee I(C_2)$
  - else $I(C) = I(C_1) \wedge I(C_2)$

There exist other interpolants; it is not clear which ones are the best

- interpolants represented by small formulas would be good, but we do not know efficient algorithms to find them

Interpolation-Based Algorithm for Reachability Checking

SAT solvers can give us proofs from which we efficiently compute interpolants.

We use interpolants to improve upon bounded model checking algorithm to give it a chance to prove properties without unfolding up to maximum length.

K.L. McMillan Interpolation and SAT-Based Model Checking. CAV 2003.

Bounded Model Checking (Checking Errors Along the Way)

Construct a propositional formula Tr_k such that formula is satisfiable if and only if there exist a trace of length **up to** k starting from initial state that satisfies error formula E where $FV(E) \subseteq \{s_1, \ldots, s_n\}$. $\bar{s}^i$ are state variables and $\bar{a}^i$ are inputs in step i.

$$\begin{tikzcd}
 & & \bar{a}^0 \arrow[d] & & \bar{a}^1 \arrow[d] & & & \bar{a}^k \arrow[d] & & \\
\text{Init} \arrow[r] & \bar{s}^0 \arrow[r] \arrow[rrrrrrrr, bend left] & \boxed{R_0} \arrow[r] & \bar{s}^1 \arrow[r] \arrow[rrrrrr, bend left] & \boxed{R_1} \arrow[r] & \cdots & s^{k-1} \arrow[r] \arrow[rrr, bend left] & \boxed{R_k} \arrow[r] & \bar{s}^k \arrow[r] & \text{E}
\end{tikzcd}$$

$$Tr_k \equiv Init[\bar{s} := \bar{s}^0] \wedge \left(\bigwedge_{i=0}^{k-1} R_i \right) \wedge \bigvee_{i=0}^{k} E[\bar{s} := \bar{s}^i]$$

where R_i is our transition formula, with variables renamed:

$$R_i \equiv R[(\bar{s}, \bar{a}, \bar{x}, \bar{s}') := (\bar{s}^i, \bar{a}^i, \bar{x}^i, \bar{s}^{i+1})]$$

Review: Reachability Checking using Precise Image Computation

Remember $post(X) = \bar{r}[X] = X \bullet \bar{r} = \{s' \mid \exists s.\ s \in X \land (s,s') \in \bar{r}\}$ where $\bar{r}$ is transition without inputs (see first lecture). We can compute reachable states and check if error is reachable by calling reachable(Init):

```
def reachable(CS): Boolean =
  val CS' = CS ∪ post(CS)
  if CS' ∩ error ≠ ∅ then true // could also just check at the end
  else if CS' ⊆ CS then false // same as checking CS'=CS
  else reachable(CS')
```

Let CS, E be formulas over $\bar{s}$ variables. R is over $\bar{s}, \bar{a}, \bar{x}, \bar{s}'$.
We can define the operations using formulas:

$CS \cup post(CS)$	$CS \lor (\exists \bar{s}.\ CS \land (\exists \bar{a}, \bar{x}. R))[\bar{s}' := \bar{s}]$
$CS' \cap error \neq \emptyset$	$sat(CS' \land E)$
$CS' \subseteq CS$	$unsat(CS' \land \neg CS)$ (that is, $CS' \models CS$)

To represent quantifiers, either expand them, or, better, use BDDs

Finding Invariants Instead of Computing Reachable States

Problems with computing reachable states:

- resulting images may be complex (large BDDs or other formulas)
- it may take many steps to compute the set of all reachable states (some behaviors manifest only in long traces)

Observation

A formula that says nothing about some boolean variables of the state both denotes a larger set of states *and* is smaller as a formula.

Instead of formula $F_{Interesting} \land F_{Complicated}$ we prefer $F_{Interesting}$
Say we have an adder and a multiplier with accumulator in our system, but our property involves only the state of the adder and is not affected by previous states of the multiplier. Reachable states may contain complex descriptions of the multiplier (it may compute e.g. factorial). Yet we may not need these details to prove our property.
Idea: find **larger sets** than $post^i(Init)$ that can be described by **smaller formulas**.

Approximate Image for Proving Property

Suppose we have operators $post_i^{\#} : 2^S \rightarrow 2^S$ such that, for all $X \subseteq S$,

$$post(X) \subseteq post_i^{\#}(X)$$

We say each $post_i^{\#}$ over-approximates $post$.
Note that $X \subseteq Y$ implies $post(X) \subseteq post(Y)$.
Consider the sequence:

$$\begin{array}{c} Init \\ post(Init) \\ post^2(Init) \\ \dots \\ post^n(Init) \end{array} \qquad \begin{array}{c} Init \\ post_1^{\#}(Init) \\ post_2^{\#}(post_1^{\#}(Init)) \\ \dots \\ post_n^{\#}(\dots post_2^{\#}(post_1^{\#}(Init))\dots) \end{array}$$

What relationship holds between corresponding elements of these sequences?

Approximate Image for Proving Property

Suppose we have operators $post_i^{\#} : 2^S \rightarrow 2^S$ such that, for all $X \subseteq S$,

$$post(X) \subseteq post_i^{\#}(X)$$

We say each $post_i^{\#}$ over-approximates $post$.
Note that $X \subseteq Y$ implies $post(X) \subseteq post(Y)$.
Consider the sequence:

$$\begin{array}{ccc} Init & \quad & Init \\ post(Init) & & post_1^{\#}(Init) \\ post^2(Init) & & post_2^{\#}(post_1^{\#}(Init)) \\ \ldots & & \ldots \\ post^n(Init) & & post_n^{\#}(\ldots post_2^{\#}(post_1^{\#}(Init))\ldots) \end{array}$$

What relationship holds between corresponding elements of these sequences?
$post(post^i(Init)) \subseteq post(post_i^{\#}(\ldots)) \subseteq post_{i+1}^{\#}(post_i^{\#}(\ldots))$

Proving that Errors are Unreachable using Approximate post

We move from checking exact reachability with reachable(Init)

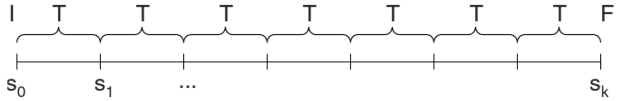
```
def reachable(CS): Boolean =
  val CS' = CS ∪ post(CS)
  if CS' ∩ error ≠ ∅ then true // could also just check at the end
  else if CS' ⊆ CS then false // same as checking CS'=CS
  else reachable(CS')
```

To checking approximate reachability with maybeReachable(Init, Init,1)

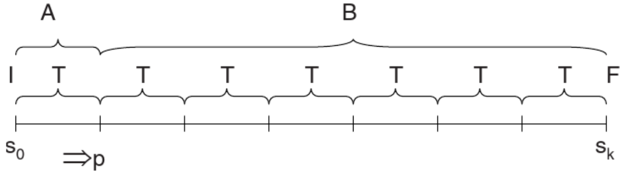
```
def maybeReachable(CS, Pn,i): Boolean =
  val Pn' = $post_i^\#$(Pn) // $post_n^\#(\ldots post_2^\#(post_1^\#(Init))\ldots)$
  val CS' = CS ∪ Pn'
  if CS' ∩ error ≠ ∅ then true // maybe. Next time use better $post^\#$
  else if CS' ⊆ CS then false // for sure not reachable. system safe!
  else maybeReachable(CS',Pn',i+1)
```

Approximate Image Using Interpolation

Take bounded model checking query for some path length k:



Compute interpolant between the formula $A \approx Init \wedge R$, and the rest, B



Use this interpolant as the result of $post_i^{\#}$. Two different kinds of iteration:

- move from a set of states Pn to $post_i^{\#}(Pn)$
  false answers mean we proved non-reachability
- increase path length k that determines B
  For k large enough, true answer becomes definitive, too!

Definition of A and B

$$\begin{tikzcd}
 & & \bar{a}^0 \arrow[d] & & \bar{a}^1 \arrow[d] & & \bar{a}^k \arrow[d] & & \\
\text{Init} \arrow[r] & \bar{s}^0 \arrow[r] \arrow[rrrrrrr, bend left] & \boxed{R_0} \arrow[r] & \bar{s}^1 \arrow[r] \arrow[rrrrr, bend left] & \boxed{R_1} \arrow[r] & \cdots\ s^{k-1} \arrow[r] \arrow[rrr, bend left] & \boxed{R_k} \arrow[r] & \bar{s}^k \arrow[r] & \text{E}
\end{tikzcd}$$

$$\underbrace{Init[\bar{s} := \bar{s}^0] \ \wedge\ R_0}_{A} \ \wedge \underbrace{\left(\bigwedge_{i=1}^{k-1} R_i \right) \ \wedge\ \bigvee_{i=0}^{k} E[\bar{s} := \bar{s}^i]}_{B}$$

State variables in R_i are $\bar{s}^i$ and $\bar{s}^{i+1}$.
Properties of (A, B) interpolant H:

- Which state variables does (A, B) interpolant H contain?

Definition of A and B

$$\begin{tikzcd}
 & & \bar{a}^0 \arrow[d] & & \bar{a}^1 \arrow[d] & & \bar{a}^k \arrow[d] & & \\
\text{Init} \arrow[r] & \bar{s}^0 \arrow[r] \arrow[rrrrrrr, bend left] & \boxed{R_0} \arrow[r] & \bar{s}^1 \arrow[r] \arrow[rrrrr, bend left] & \boxed{R_1} \arrow[r] & \cdots\ s^{k-1} \arrow[r] \arrow[rrr, bend left] & \boxed{R_k} \arrow[r] & \bar{s}^k \arrow[r] & \text{E}
\end{tikzcd}$$

$$\underbrace{Init[\bar{s} := \bar{s}^0] \ \wedge \ R_0}_{A} \ \wedge \underbrace{\left(\bigwedge_{i=1}^{k-1} R_i\right) \ \wedge \ \bigvee_{i=0}^{k} E[\bar{s} := \bar{s}^i]}_{B}$$

State variables in R_i are $\bar{s}^i$ and $\bar{s}^{i+1}$.
Properties of (A, B) interpolant H:

- Which state variables does (A, B) interpolant H contain? Answer: $\bar{s}^1$
- $A \models H$ becomes:

Definition of A and B

$$\begin{tikzcd}
 & & \bar{a}^0 \arrow[d] & & \bar{a}^1 \arrow[d] & & \bar{a}^k \arrow[d] & & \\
\text{Init} \arrow[r] & \bar{s}^0 \arrow[r] \arrow[rrrrrrr, bend left] & \boxed{R_0} \arrow[r] & \bar{s}^1 \arrow[r] \arrow[rrrrr, bend left] & \boxed{R_1} \arrow[r] & \cdots\ s^{k-1} \arrow[r] \arrow[rrr, bend left] & \boxed{R_k} \arrow[r] & \bar{s}^k \arrow[r] & \text{E}
\end{tikzcd}$$

$$\underbrace{Init[\bar{s} := \bar{s}^0] \ \wedge \ R_0}_{A} \ \wedge \underbrace{\left(\bigwedge_{i=1}^{k-1} R_i \right) \ \wedge \ \bigvee_{i=0}^{k} E[\bar{s} := \bar{s}^i]}_{B}$$

State variables in R_i are $\bar{s}^i$ and $\bar{s}^{i+1}$.
Properties of (A, B) interpolant H:

- Which state variables does (A, B) interpolant H contain? Answer: $\bar{s}^1$
- $A \models H$ becomes: $Init[\bar{s} := \bar{s}^0] \ \wedge \ R_0 \models H$
- $B \models \neg H$ becomes:

Definition of A and B

$$\begin{tikzcd}
 & & \bar{a}^0 \arrow[d] & & \bar{a}^1 \arrow[d] & & \bar{a}^k \arrow[d] & & \\
\text{Init} \arrow[r] & \bar{s}^0 \arrow[r] \arrow[rrrrrrr, bend left] & \boxed{R_0} \arrow[r] & \bar{s}^1 \arrow[r] \arrow[rrrrr, bend left] & \boxed{R_1} \arrow[r] & \cdots\ s^{k-1} \arrow[r] \arrow[rrr, bend left] & \boxed{R_k} \arrow[r] & \bar{s}^k \arrow[r] & \text{E}
\end{tikzcd}$$

$$\underbrace{Init[\bar{s} := \bar{s}^0] \;\wedge\; R_0}_{A} \;\wedge \underbrace{\left(\bigwedge_{i=1}^{k-1} R_i\right) \;\wedge\; \bigvee_{i=0}^{k} E[\bar{s} := \bar{s}^i]}_{B}$$

State variables in R_i are $\bar{s}^i$ and $\bar{s}^{i+1}$.
Properties of (A, B) interpolant H:

- Which state variables does (A, B) interpolant H contain? Answer: $\bar{s}^1$
- $A \models H$ becomes: $Init[\bar{s} := \bar{s}^0] \;\wedge\; R_0 \models H$
- $B \models \neg H$ becomes: $unsat(H \wedge B)$, which prevents H from denoting too large set

Interpolant is Over-Approximation

$A \models H$ condition:

$$Init[\bar{s} := \bar{s}^0] \ \wedge \ R_0 \models H$$

can be written (by moving quantifiers to formulas where they may appear) as:

$$\underbrace{\exists \bar{s}^0.\ (Init[\bar{s} := \bar{s}^0] \wedge \exists \bar{x}, \bar{a}^0.R_0)}_{Init \bullet \bar{r}} \underbrace{\models}_{\subseteq} H$$

Define $post_1^{\#}(Init) \equiv H$. Then $post(Init) \subseteq post_1^{\#}(Init)$
Next, we treat $H[\bar{s}^1 := \bar{s}^0]$ as our next Init and repeat:

$$\underbrace{H[\bar{s}^1 := \bar{s}^0] \ \wedge \ R_0}_{A^1} \ \wedge \underbrace{\left(\bigwedge_{i=1}^{k-1} R_i\right) \ \wedge \ \bigvee_{i=0}^{k} E[\bar{s} := \bar{s}^i]}_{B_k}$$

and find the interpolant for (A^1, B_k).

Algorithm

Nested loop changes interpolation pair (A, B_k): inner loop (i) changes A, outer B_k:

```
iReachable(Init, 0, 1)
def iReachable(CS, i, k): Boolean = // i is not really used
  iSAT($CS \wedge R_0$, $B_k$) match
    case SAT(e) ⇒
      if CS=Init then true // reachable from Init, e gives trace
      else iReachable(Init,0,k+1) // restart with longer $B_k$
    case UNSAT(interpolant) ⇒
      val H' = interpolant[$\bar{s}^1$:=$\bar{s}$] // H' over-approximates $post^i(Init)$
      val CS' = CS ∨ H'
      if unsat($H' \wedge \neg CS$) then false // CS'=CS, for sure not reachable
      else iReachable(CS',i+1, k) // keep iterating $post^\#$
```

Properties:

- sound for "true" because it reduces to bounded model checking (obvious)
- sound for "false" answer because CS contains $\bigcup_{j=0}^{i} post^j(I)$
- terminates by $k \leq 2^n$ for n propositional variables in transition system

Soundness for false (Unreachability): CS is Over-Approximation

We prove by induction on i that, for CS in iteration i, we have

$$CS_{set} \supseteq \bigcup_{j=0}^{i} post^j(Init_{set})$$

where for a formula F over $\bar{s}$ we define the corresponding set of states
$F_{set} = \{\bar{x} \mid [\![F]\!]_{[\bar{s} \mapsto \bar{x}]}\}$

For $i = 0$, both sides equal $Init_{set}$.
Let property hold for i. By definition $CS \land R_0 \models interpolant$, so $post(CS_{set}) \subseteq H'_{set}$ and:

$$CS'_{set} = CS_{set} \cup H'_{set} \supseteq CS_{set} \cup post(CS_{set}) \supseteq CS_{set} \cup post(\bigcup_{j=0}^{i} post^j(Init_{set}))$$

where the last set equals $\bigcup_{j=0}^{i+1} post^j(Init_{set})$ by definition of $post$.

Soundness for false: Over-Approximation Consequence

For large enough i in a finite-state system, the value for $i+1$ is same as for i in

$$CS_{set} \supseteq \bigcup_{j=0}^{i} post^{j}(Init_{set})$$

Similarly, CS can only grow a finite number of steps because it is within the set of all states.

Thus, sequences stabilize and both could be extended to the point where they are both stable. At this point, CS has value it reached when it first stabilized, whereas the right-hand side is the set of reachable states. Thus, stabilized CS contains the set of reachable states.

If CS does not intersect error, then neither does the set of reachable states.

Termination: Path Formulas Become Weaker

For simplicity we (like the paper) assume that $\bar{r}$ is a total relation:

$$\forall s.\exists s'.(s,s') \in \bar{r}$$

This is a almost always true.
We have defined B_k as:

$$\left(\bigwedge_{i=1}^{k-1} R_i\right) \wedge \bigvee_{i=0}^{k} E[\bar{s} := \bar{s}^i]$$

Claim: Let F be formula not referring only to $\bar{s}^{k+1}, \bar{a}^k, \bar{x}^k$. Then
$F \wedge B_k \models \exists \bar{s}_{k+1}.(F \wedge B_{k+1})$

$$F \wedge B_k \quad \models \quad \exists \bar{s}_{k+1}.\left(F \wedge \left(\bigwedge_{i=1}^{k-1} R_i\right) \wedge R_k \ \wedge \ \left(\bigvee_{i=0}^{k} E[\bar{s} := \bar{s}^i] \vee E[\bar{s} := \bar{s}^{k+1}]\right)\right)$$

Proof: given e where $[\![B_k]\!]_e = 1$, define e' as e but with $e'(\bar{s}^{k+1}, \bar{a}^k, \bar{x}^k)$ such that $(e'(\bar{s}^k), e'(\bar{s}^{k+1})) \in \bar{r}$ by totality of $\bar{r}$ and $[\![R_k]\!]_{e'} = 1$ by definition of $\bar{r}$. Then, $[\![F \wedge \bigwedge_{i=1}^{k-1} R_i]\!]_{e'} = 1$ and $[\![\bigvee_{i=0}^{k} E[\bar{s} := \bar{s}^i]]\!]_{e'} = 1$ as these formulas do not refer to variables whose values differ between e' and e and they are true in e.

Termination: Interpolants Become Tighter

Compare interpolants H for (A, B_k) vs (A, B_{k+1}). We get $A \models H$ in both cases for lower bound. For upper bound, we get $H \models \neg B_k$ and $H \models \neg B_{k+1}$, or, equivalently,

$$\mathit{unsat}(H \wedge B_k) \ \text{ vs } \quad \mathit{unsat}(H \wedge B_{k+1})$$

Every satisfying assignment to $H \wedge B_k$ can be extended to one for $H \wedge B_{k+1}$, so $\mathit{unsat}(H \wedge B_{k+1})$ imposes a tigher upper bound on H.
$\mathit{unsat}(H \wedge B_k)$ means there are no paths **up to** length k starting in H and reaching error state. Larger k is a stronger condition on H.

H contains only $\bar{s}^1$ variables.

Termination

$unsat(H \wedge B_k)$ means $H \models \neg B_k$ and is the same as

$$H \models \forall \bar{y}. \neg B_k$$

where $\bar{y}$ denotes all variables except $\bar{s}^1$.
Let U_k be $\exists \bar{y}. B_k$. Then $H \models \neg U_k$. U_k denotes the property of $\bar{s}^1$ that there exists a path of length up to k that reaches a state satisfying E.
Let D be the maximum of lengths of simple paths (without repeated states) that terminate at a state satisfying E.
Then U_k for all $k \geq D$ are equivalent and denote all states that can reach E.
For $k \geq D$, the upper bound on interpolant becomes the same statement saying that a state cannot reach E. For such k, if we do not get SAT answer for $i = 0$, we can not get SAT answer later either: a SAT answer for some $i > 0$ implies that error is reachable in $k + 1$ steps, which is an equivalent condition that it is reachable in k steps, which was checked to be false in step $i - 1$. Thus, the answers to iSAT are always UNSAT and the case CS=CS' must be reached.