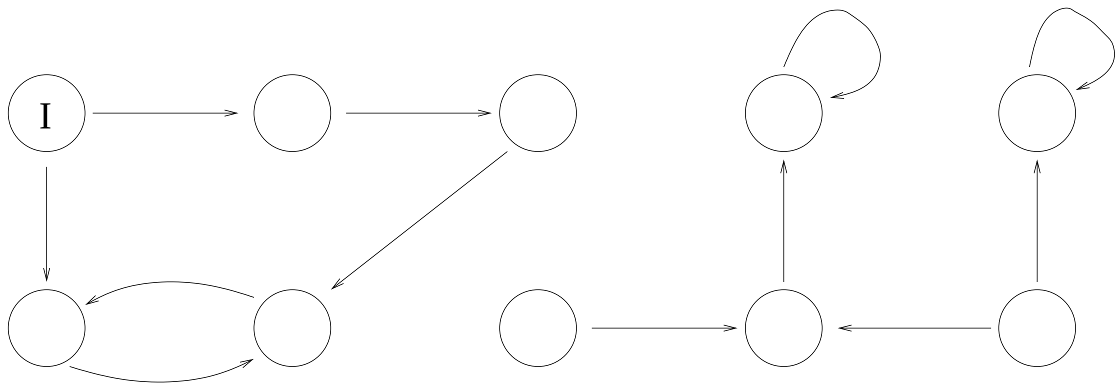
SAT based Abstraction-Refinement using ILP and Machine Learning Techniques

Edmund Clarke Anubhav Gupta
James Kukula Ofer Strichman

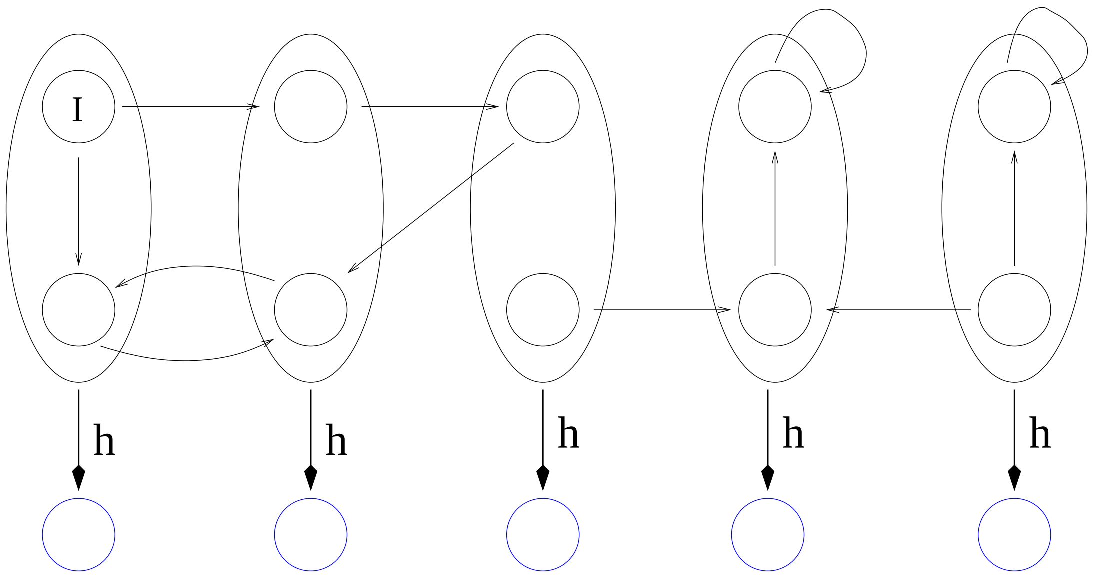
Abstraction in Model Checking



- Set of variables $V = \{x_1, \ldots, x_n\}$.
- Set of states $S = D_{x_1} \times \cdots \times D_{x_n}$.
- Set of initial states $I \subseteq S$.
- Set of transitions $R \subseteq S \times S$.
- Transition system $M = (S, I, R)$.

Abstract Model

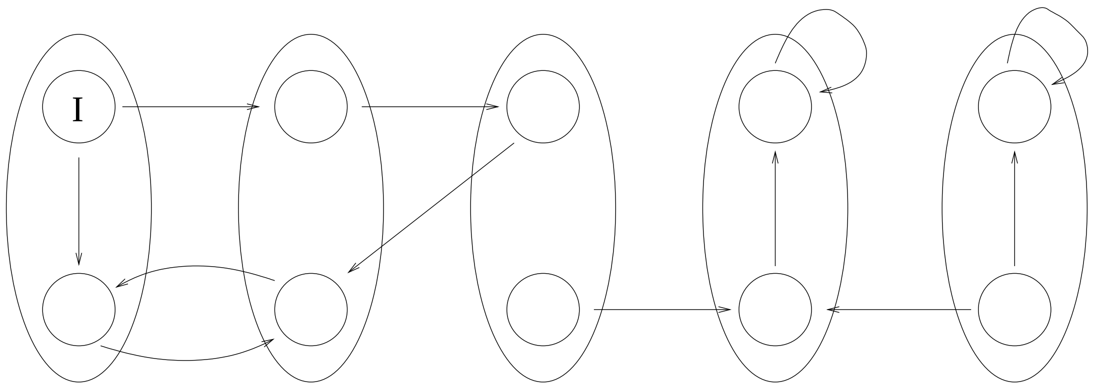
Abstraction Function $h : S \longrightarrow \widehat{S}$ $\widehat{M} = (\widehat{S}, \widehat{I}, \widehat{R})$

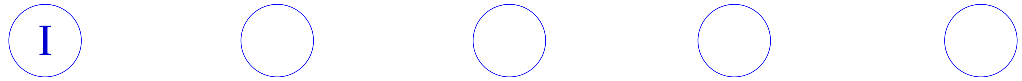


$$\widehat{S} = \{\hat{s} \mid \exists s.\ s \in S \wedge h(s) = \hat{s}\}$$

Abstract Model

Abstraction Function $h : S \longrightarrow \widehat{S}$ $\widehat{M} = (\widehat{S}, \widehat{I}, \widehat{R})$

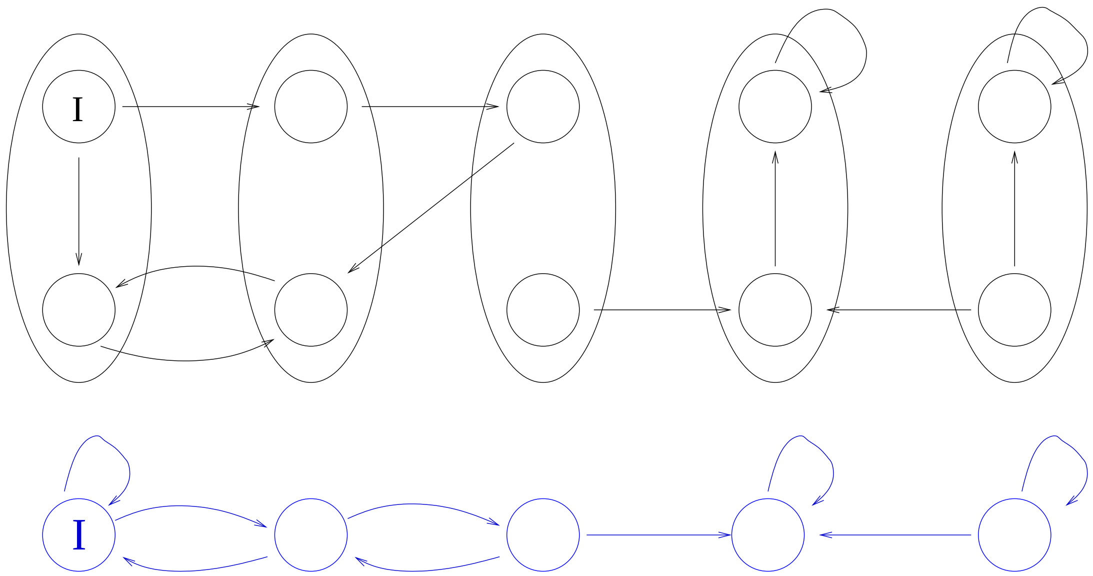




$$\widehat{I} = \{\widehat{s} \mid \exists s.\ I(s) \wedge h(s) = \widehat{s}\}$$

Abstract Model

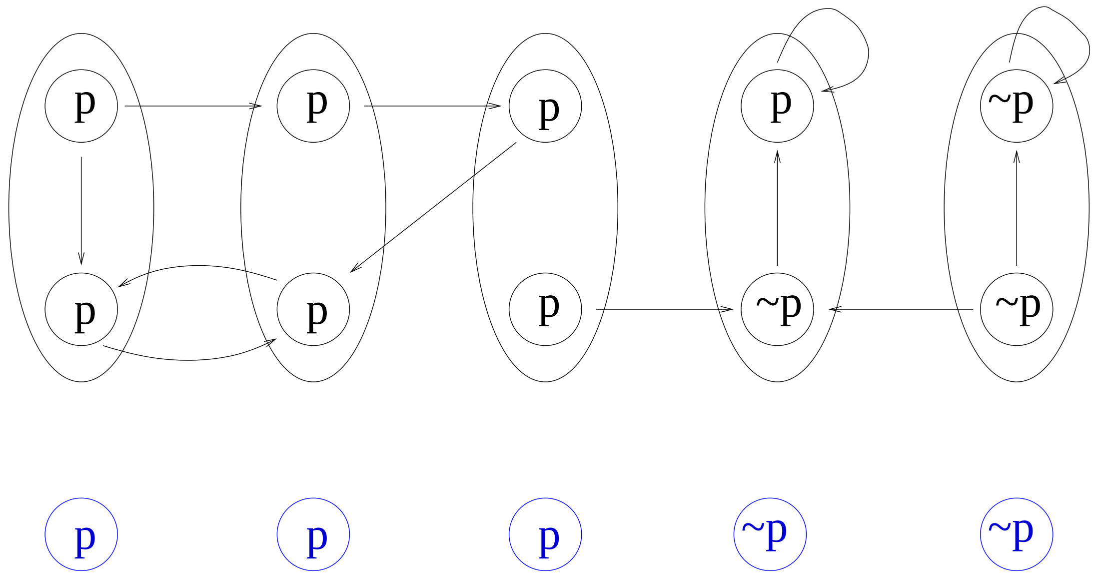
Abstraction Function $h : S \rightarrow \widehat{S}$ $\widehat{M} = (\widehat{S}, \widehat{I}, \widehat{R})$



$$\widehat{R} = \{(\widehat{s}_1, \widehat{s}_2) \mid \exists s_1.\, \exists s_2.\, R(s_1, s_2) \wedge h(s_1) = \widehat{s}_1 \wedge h(s_2) = \widehat{s}_2\}$$

Model Checking

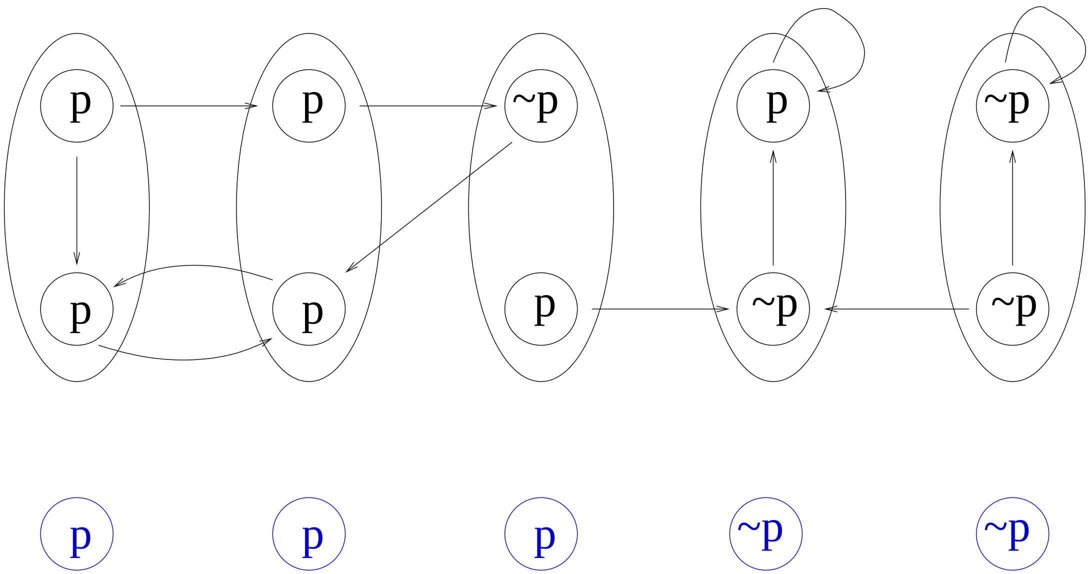
- $\mathbf{AG}p$, p is a non-temporal propositional formula
- p respects h if for all $s \in S$, $h(s) \models p \Rightarrow s \models p$



p respects h

Model Checking

- $\mathbf{AG}p$, p is a non-temporal propositional formula
- p respects h if for all $s \in S$, $h(s) \models p \Rightarrow s \models p$

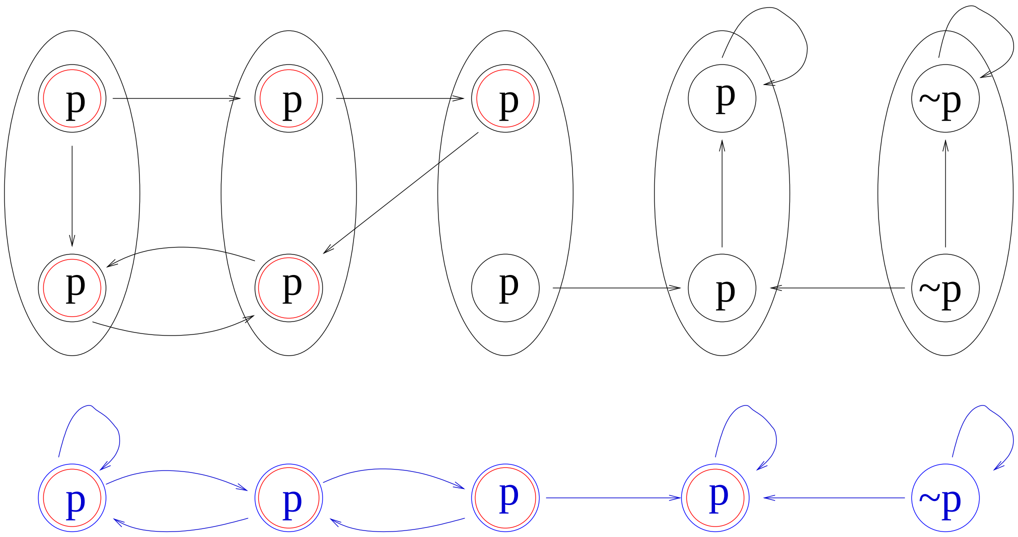


p does not respect h

Preservation Theorem

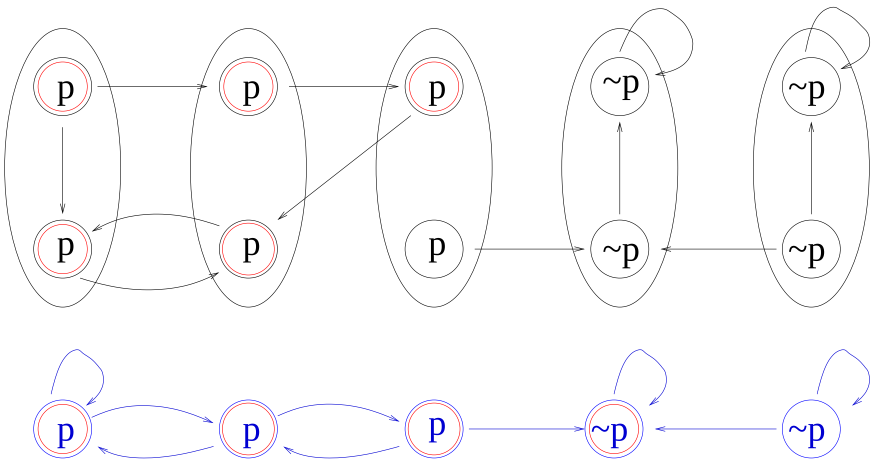
Let $\widehat{M}$ be an abstraction of M corresponding to the abstraction function h, and p be a propositional formula that respects h. Then

$$\widehat{M} \models \mathbf{AG}p \Rightarrow M \models \mathbf{AG}p$$



Converse of Preservation Theorem

$$\widehat{M} \not\models \mathbf{AG}p \not\Rightarrow M \not\models \mathbf{AG}p$$

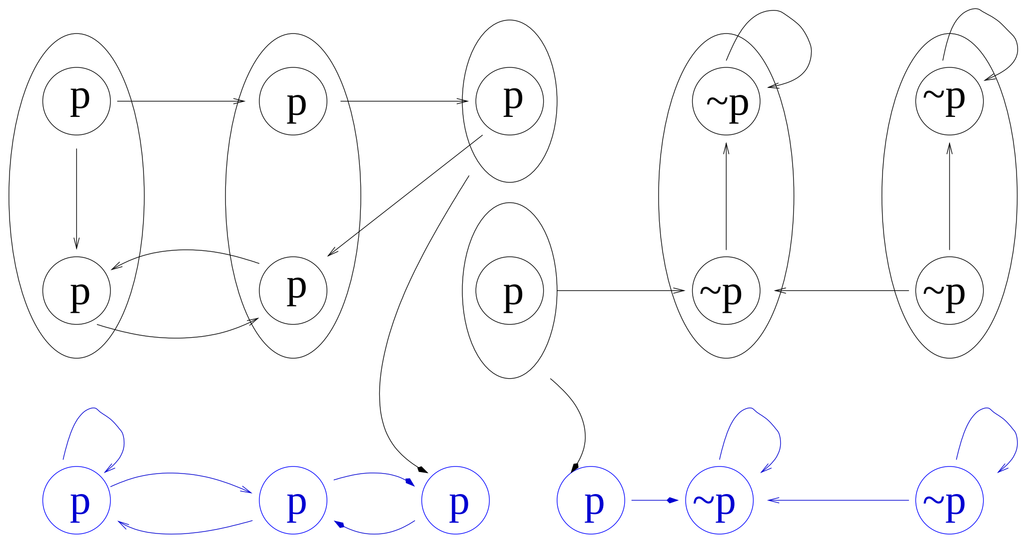


Counterexample is spurious. Abstraction is too coarse.

Refinement

h' is a refinement of h if

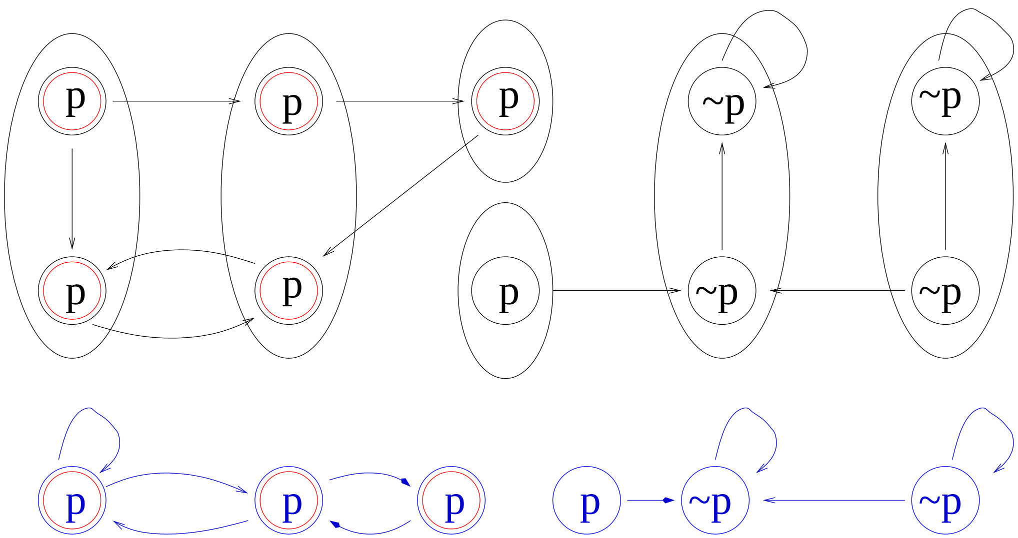
1. $\forall\, s_1, s_2 \in S$, $h'(s_1) = h'(s_2)$ implies $h(s_1) = h(s_2)$.
2. $\exists\, s_1, s_2 \in S$ such that $h(s_1) = h(s_2)$ and $h'(s_1) \neq h'(s_2)$.



Refinement

h' is a refinement of h if

1. $\forall\ s_1, s_2 \in S$, $h'(s_1) = h'(s_2)$ implies $h(s_1) = h(s_2)$.
2. $\exists\ s_1, s_2 \in S$ such that $h(s_1) = h(s_2)$ and $h'(s_1) \neq h'(s_2)$.



Abstraction-Refinement

1. Generate an initial abstraction function h.
2. Build abstract machine $\widehat{M}$ based on h. Model check $\widehat{M}$. If $\widehat{M} \models \varphi$, then $M \models \varphi$. Return TRUE.
3. If $\widehat{M} \not\models \varphi$, check the counterexample on the concrete model. If the counterexample is real, $M \not\models \varphi$. Return FALSE.
4. Refine h, and go to step 2.

Abstraction Function

- Partition variables V into visible($\mathcal{V}$) and invisible($\mathcal{I}$) variables.
  $\mathcal{V} = \{v_1, \ldots, v_k\}$.
- The partitioning defines our abstraction function $h : S \rightarrow \widehat{S}$. The set of abstract states is

$$\widehat{S} = D_{v_1} \times \cdots \times D_{v_k}$$

  and the abstraction functions is

$$h(s) = (s(v_1) \ldots s(v_k))$$

x1	x2	x3	x4
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1

x1	x2
0	0

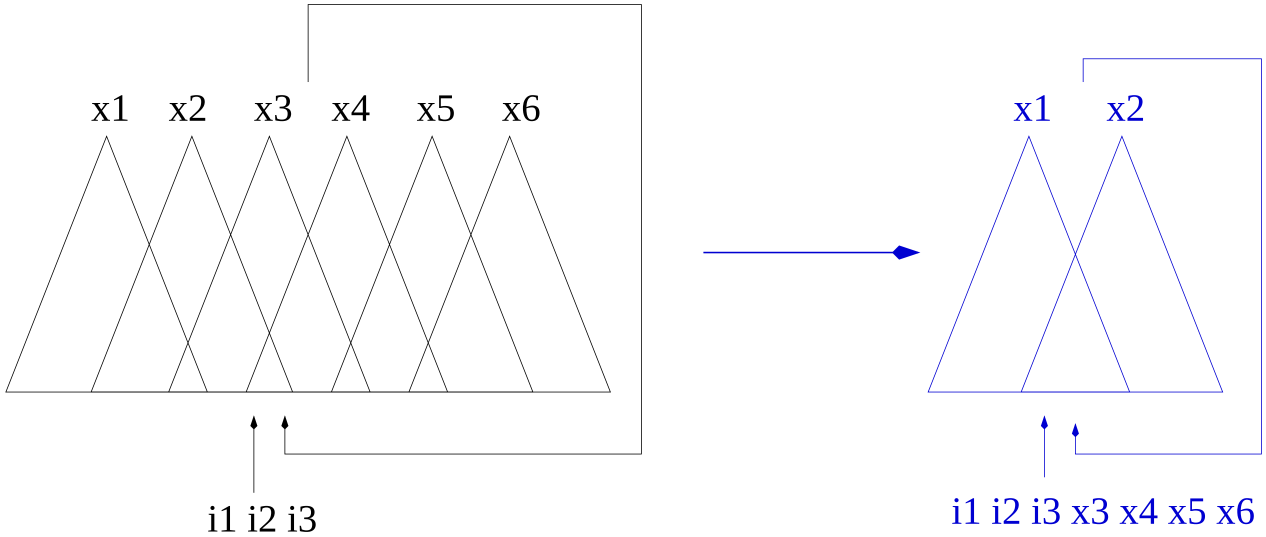
- Refinement : Move variables from $\mathcal{I}$ to $\mathcal{V}$.

Building Abstract Model

$\hat{M}$ can be computed efficiently if R is in functional form, e.g. sequential circuits.

$R(s, s') = \exists i (\bigwedge_{j=1}^{m} x'_j = f_{x_j}(s, i))$

$\hat{R}(\hat{s}, \hat{s}') = \exists s^{\mathcal{I}} \exists i (\bigwedge_{x_j \in \mathcal{V}} \hat{x}'_j = f_{x_j}(\hat{s}, s^{\mathcal{I}}, i))$



Checking the Counterexample

- Counterexample : $\langle \hat{s}_1, \hat{s}_2, \ldots \hat{s}_m \rangle$
- Set of concrete paths for counterexample :

$$\psi_m = \{\langle s_1 \ldots s_m \rangle \mid I(s_1) \wedge \bigwedge_{i=1}^{m-1} R(s_i, s_{i+1}) \wedge \bigwedge_{i=1}^{m} h(s_i) = \hat{s}_i\}$$

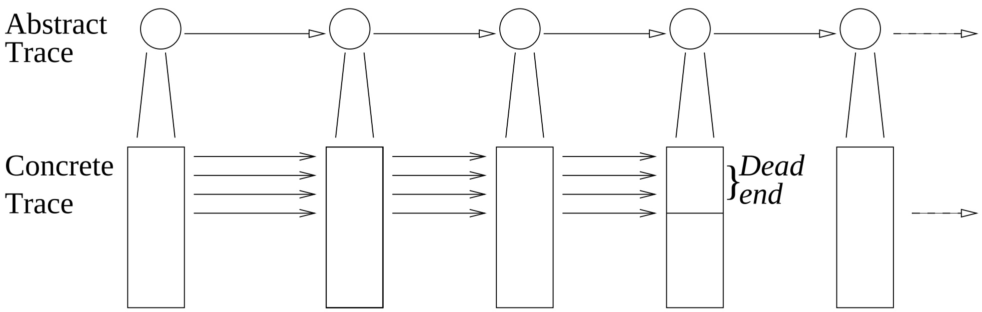
- The right-most conjunct is a restriction of the visible variables to their values in the counterexample.
- Counterexample is spurious $\iff$ ψ_m is empty.
- Solve ψ_m with a SAT solver.

Checking the Counterexample

- Similar to BMC formulas, except
  - Path restricted to counterexample.
  - Also restrict values of (original) inputs that are assigned by counterexample.
- If ψ_m is satisfiable we found a real bug.
- If ψ_m is unsatisfiable, refine.

Refinement

- Find largest index f (failure index), $f < m$ such that ψ_f is satisfiable.
- The set D of all states d_f such that there is a concrete path $\langle d_1 ... d_f \rangle$ in ψ_f is called the set of deadend states.



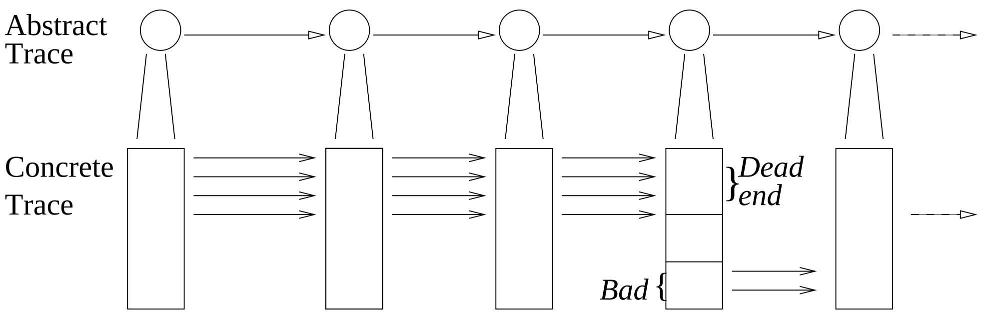
- No concrete transition from D to a concrete state in the next abstract state.

Refinement

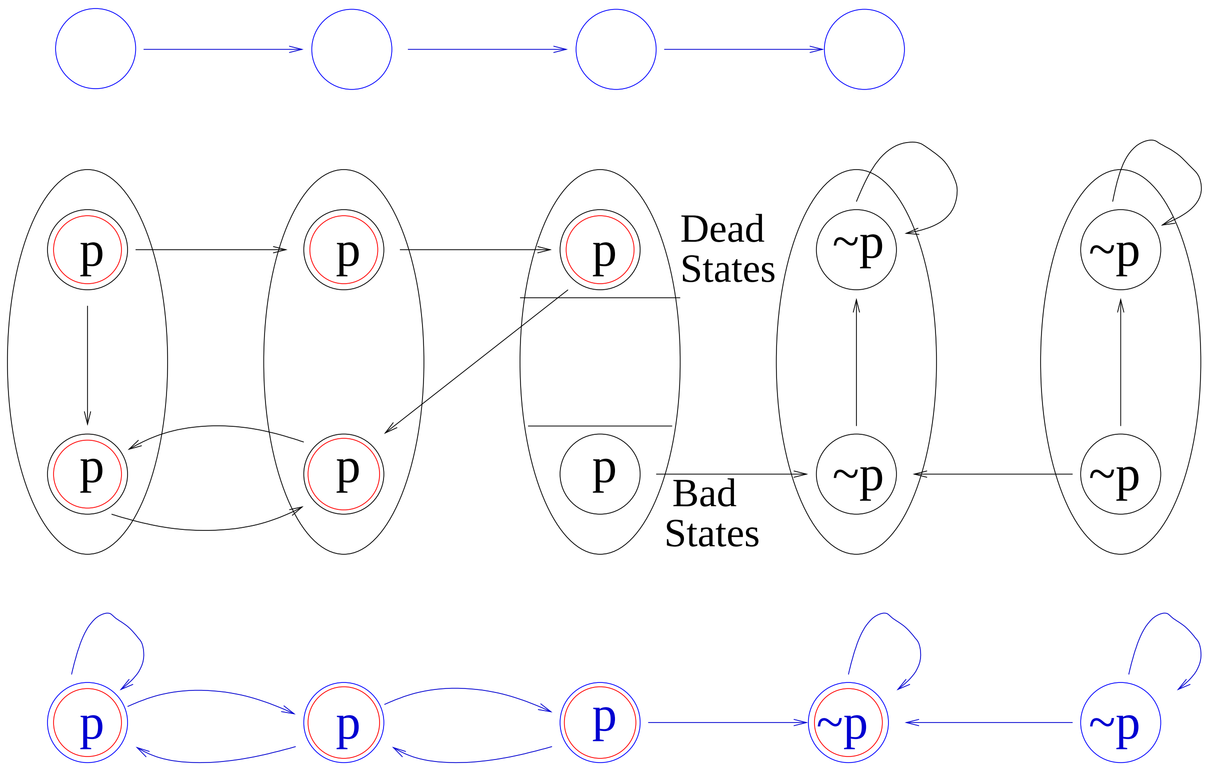
- Since there is an abstract transition from $\widehat{s}_f$ to $\widehat{s}_{f+1}$, there is a non-empty set of transitions ϕ_f from $h^{-1}(\widehat{s}_f)$ to $h^{-1}(\widehat{s}_{f+1})$.

$$\phi_f = \{\langle s_f, s_{f+1}\rangle \mid R(s_f, s_{f+1}) \wedge h(s_f) = \widehat{s}_f \wedge h(s_{f+1}) = \widehat{s}_{f+1}\}$$

- The set B of all states b_f such that there is a transition $\langle b_f, b_{f+1}\rangle$ in ϕ_f is called the set of bad states.



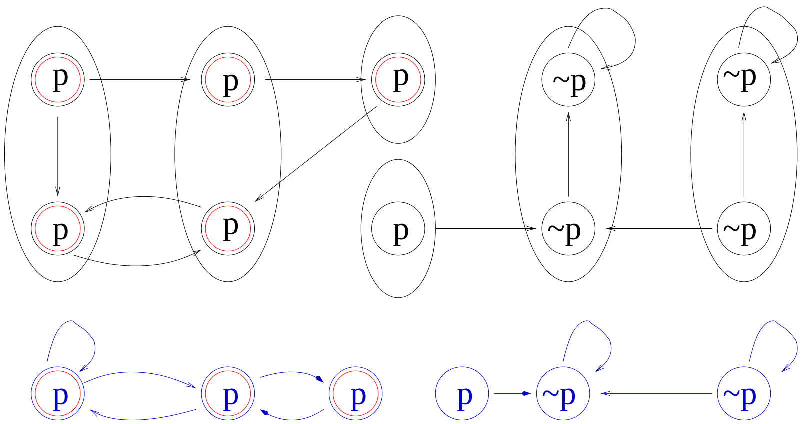
Refinement



Refinement

- There is a spurious transition from $\widehat{s}_f$ to $\widehat{s}_{f+1}$.
- Spurious transition because D and B lie in the same abstract state.
- Refinement : Put D and B is separate abstract states.

$$\forall d \in D, \forall b \in B \; (h'(d) \neq h'(b))$$



Refinement as Separation

Let $S = \{s_1 ... s_m\}$ and $T = \{t_1 ... t_n\}$ be two sets of states (binary vectors) of size l, representing assignments to a set of variables W, $|W| = l$.

(The state separation problem)
Find a minimal set of variables $U = \{u_1 ... u_k\}$, $U \subseteq W$, such that for each pair of states (s_i, t_j), $1 \leq i \leq m$, $1 \leq j \leq n$, there exists a variable $u_r \in U$ such that $s_i(u_r) \neq t_j(u_r)$.

Let H denote the separating set for D and B. The refinement h' is obtained by adding H to $\mathcal{V}$.

Proof : Since H separates D and B, for all $d \in D$, $b \in B$ there exists $u \in H$ s.t. $d(u) \neq b(u)$. Hence, $h(d) \neq h(b)$.

Refinement as Separation and Learning

- For systems of realistic size,
  - It is not possible to generate D and B, either explicitly or symbolically.
  - Computationally expensive to separate large D and B.
- Generate samples for D(denoted S_D) and B(denoted S_B) and try to infer the separating variables from the samples.
- State of the art SAT solvers like Chaff can generate many samples in a short amount of time.
- Our algorithm is complete because a counterexample will eventually be eliminated in subsequent iterations.

Separation using Integer Linear Programming

Separating S_D from S_B as an Integer Linear Programming (ILP) problem:

$$\text{Min} \sum_{i=1}^{|\mathcal{I}|} v_i$$

$$\text{subject to:} \quad (\forall s \in S_D)\ (\forall t \in S_B) \sum_{\substack{1 \leq i \leq |\mathcal{I}|,\\ s(v_i) \neq t(v_i)}} v_i \geq 1$$

- $v_i = 1$ if and only if v_i is in the separating set.
- One constraint per pair of states, stating that at least one of the variables that separates the two states should be selected.

Example

$$s_1 = (0, 1, 0, 1) \qquad t_1 = (1, 1, 1, 1)$$
$$s_2 = (1, 1, 1, 0) \qquad t_2 = (0, 0, 0, 1)$$

Min $\sum_{i=1}^{4} v_i$

subject to:

$v_1 + v_3$	≥ 1	/* Separating s_1 from t_1 */
v_2	≥ 1	/* Separating s_1 from t_2 */
v_4	≥ 1	/* Separating s_2 from t_1 */
$v_1 + v_2 + v_3 + v_4$	≥ 1	/* Separating s_2 from t_2 */

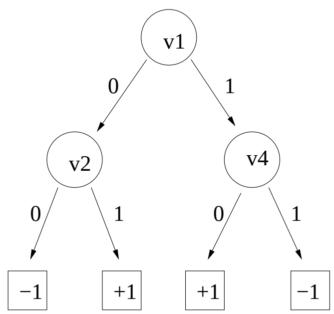
Optimal value of the objective function is 3, corresponding to one of the two optimal solutions (v_1, v_2, v_4) and (v_3, v_2, v_4).

Separation using Decision Tree Learning

- ILP-based separation:
  - Minimal separation set
  - Computationally expensive
- Decision Tree Learning based separation:
  - Non optimal
  - Computationally efficient

Decision Tree Learning

- Input : Set of examples with classification.
  - Each example assigns values to a set of attributes.
- Output : Decision Tree
  - Each internal node is a test on some attribute.
  - Each leaf corresponds to a classification.



Separation using Decision Tree Learning

Separating S_D from S_B as a Decision Tree Learning problem:

- Attributes correspond to the invisible variables.
- The classifications are $+1$ and -1, corresponding to S_D and S_B, respectively.
- The examples are S_D labeled $+1$, and S_B labeled -1.

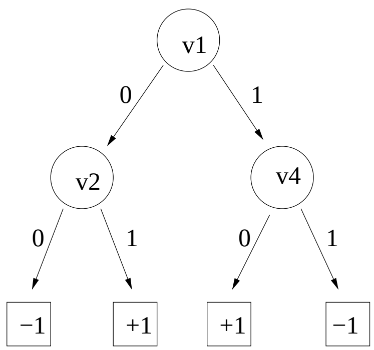
Separating set : All the variables present at an internal nodes of the decision tree.

Proof: Let $d \in S_D$ and $b \in S_B$. The decision tree will classify d as $+1$ and b as -1. So, there exists a node n in the decision tree, labeled with a variable v, such that $d(v) \neq b(v)$. By construction, v lies in the output set.

Example

$$s_1 = (0,1,0,1) \qquad t_1 = (1,1,1,1)$$
$$s_2 = (1,1,1,0) \qquad t_2 = (0,0,0,1)$$

$$E = (((0,1,0,1),+1),((1,1,1,0),+1),((1,1,1,1),-1),((0,0,0,1),-1)$$



Separating set : $\{v_1, v_2, v_4\}$

Decision Tree Learning Algorithm

$DecTree(Examples, Attributes) \quad ID3\ Algorithm$

1. Create a $Root$ node for the tree.
2. If all examples are classified the same, return $Root$ with this classification.
3. Let $A = BestAttribute(Examples, Attributes)$. Label $Root$ with attribute A.
4. Let $Examples_0$ and $Examples_1$ be subsets of $Examples$ having values 0 and 1 for A, respectively.
5. Add a 0 branch to the $Root$ pointing to subtree generated by $Dectree(Examples_0, Attributes - \{A\})$.

6. Add a 1 branch to the $Root$ pointing to subtree generated by $Dectree(Examples_1, Attributes - \{A\})$.
7. return $Root$.

The $BestAttribute$ procedure returns an attribute (which is a variable in our case) that causes the maximum reduction in entropy if the set is partitioned according to this variable.

Efficient Sampling

- Direct search towards samples that contain more information.
- Iterative Algorithm.
- At each iteration, the algorithm finds new samples that are not separated by the current separating set.
- Let $SepSet$ denote the separating set for the current set of samples. New samples that are not separated by $SepSet$ are computed by solving

$$\Phi(SepSet) \doteq \psi_f \wedge \phi'_f \wedge \bigwedge_{v_i \in SepSet} v_i = v'_i$$

Efficient Sampling

$SepSet = \emptyset$;

$i = 0$;

repeat forever {

If $\Phi(SepSet)$ is satisfiable, derive d_i and b_i from solution; else exit;

$SepSet = Separating\ Set\ for\ \{\bigcup_{j=0}^{i}\{d_j\},\ \bigcup_{j=0}^{i}\{b_j\}\}$;

$i = i + 1$; }

Experiments

- NuSMV frontend.
- Cadence SMV.
- A public domain ILP solver.
- Chaff.

Experiments conducted on a 1.5GHz Athlon with 3Gb RAM running Linux.

We used the “IU” family of circuits, which are various abstractions of an interface control circuit from Synopsys.

Circuit	SMV		Sampling - ILP				Sampling - DTL				Eff. Samp. - DTL			
	Time	BDD(k)	Time	BDD(k)	S	L	Time	BDD(k)	S	L	Time	BDD(k)	S	L
$IU30$	0.7	116	0.1	1	0	1	0.1	1	0	1	**0.1**	1	0	1
$IU35$	0.6	149	0.1	2	0	1	0.1	2	0	1	**0.1**	2	0	1
$IU40$	1.2	225	6.3	21	3	4	0.9	18	5	6	**0.6**	11	2	3
$IU45$	37.5	2554	6.1	17	3	4	1.1	18	5	6	**0.7**	10	2	3
$IU50$	23.3	2094	19.7	100	13	14	**9.8**	90	13	14	24.0	1274	4	17
$IU55$	-	-	-	-	-	-	2072	51703	6	9	**3.0**	64	1	6
$IU60$	-	-	7.8	183	4	7	7.8	183	4	7	**4.5**	109	1	6
$IU65$	-	-	7.9	192	4	7	7.9	192	4	7	**3.8**	47	1	5
$IU70$	-	-	8.1	192	4	7	8.2	192	4	7	**3.8**	47	1	5
$IU75$	102.9	7068	32.0	142	9	10	24.5	397	13	14	**24.1**	550	2	7
$IU80$	603.7	39989	31.7	215	9	10	44.0	341	13	14	**24.1**	186	2	7
$IU85$	2832	76232	33.1	230	9	10	44.6	443	13	14	**25.2**	198	2	7
$IU90$	-	-	33.0	230	9	10	44.6	443	13	14	**25.4**	198	2	7

Circuit	SMV		Sampling - ILP				Sampling - DTL				Eff. Samp. - DTL			
	Time	BDD(k)	Time	BDD(k)	S	L	Time	BDD(k)	S	L	Time	BDD(k)	S	L
$IU30$	7.3	324	8.0	113	3	20	7.5	113	3	20	**6.5**	113	3	20
$IU35$	19.1	679	11.8	186	4	21	12.7	186	4	21	**11.0**	186	4	21
$IU40$	53.6	1100	25.9	260	6	23	19.0	207	5	22	**16.1**	207	5	22
$IU45$	226.1	6060	28.3	411	5	22	25.3	411	5	22	**22.1**	411	5	22
$IU50$	1754	25102	160.4	2046	13	32	**85.1**	605	10	27	15120	3791	7	31
$IU55$	-	-	-	-	-	-	-	-	-	-	-	-	-	-
$IU60$	-	-	-	-	-	-	-	-	-	-	-	-	-	-
$IU65$	-	-	-	-	-	-	-	-	-	-	-	-	-	-
$IU70$	-	-	-	-	-	-	-	-	-	-	-	-	-	-
$IU75$	-	-	1080	3716	21	38	586.7	1178	16	33	**130.5**	1050	5	26
$IU80$	-	-	1136	3378	21	38	552.5	1158	16	33	**153.4**	1009	5	26
$IU85$	-	-	1162	3493	21	38	581.2	1272	16	33	**167.7**	1079	5	26
$IU90$	-	-	965	3712	20	37	583.3	1271	16	33	**167.1**	1079	5	26

Conclusions and Future Work

- Our algorithm outperforms standard model checking in both execution time and memory requirements.
- Exploit criteria other than size of separating set for characterizing a good refinement.
- Explore other learning techniques.