Automatic Abstraction

Current automatic abstraction tools typically proceed as follows:

• Given a concrete program $C$, they generate an abstract program $A$ such that “$A$ simulates $C$”.
• For any $\forall$-properties $\phi$, $A \models \phi$ implies $C \models \phi$.

Limitations:

• Restricted to $\forall$-properties (no existential properties).
• $A \not\models \phi$ does not imply anything about $C$!
• Could the analysis be more precise for a comparable cost?
A Solution: use 3-Valued Models [Bruns-G99]

Use richer models $A$ that distinguish what is \textit{true}, \textit{false} and unknown ($\bot$) of $C$.

Example: partial Kripke structure (PKS) [Fitting92,Bruns-G99]

- A Kripke structure where propositions can be \textit{true}, \textit{false} or $\bot$.

Example: Modal Transition System [Larsen-Thomsen88]

- A LTS with $\xrightarrow{\text{may}}$ and $\xrightarrow{\text{must}}$ transitions such that $\xrightarrow{\text{must}} \subseteq \xrightarrow{\text{may}}$.

Example: Kripke Modal Transition System [Huth-Jagadeesan-Schmidt01]

- A PKS with $\xrightarrow{\text{may}}$ and $\xrightarrow{\text{must}}$ transitions such that $\xrightarrow{\text{must}} \subseteq \xrightarrow{\text{may}}$.

These models are all equally expressive [G-Jagadeesan03].

Other examples: extended transition systems [Milner81],...
3-Valued Temporal Logics

Reasoning about 3-valued models requires 3-valued TL.

Example: 3-valued Propositional Modal Logic \( \phi ::= p \mid \neg \phi \mid \phi_1 \land \phi_2 \mid AX \phi \)

Semantics: (extension of Kleene’s strong 3-valued PL)

\[ [(M, s) \models p] = L(s, p) \]

\[ [(M, s) \models \neg \phi] = \text{comp}([(M, s) \models \phi]) \]

where comp maps \textit{true} \mapsto \textit{false}, \textit{false} \mapsto \textit{true}, and \( \bot \mapsto \bot \)

\[ [(M, s) \models \phi_1 \land \phi_2] = \text{min}([(M, s) \models \phi_1], [(M, s) \models \phi_2]) \]

with min defined with \textit{false} \(<\bot<\textit{true} \ (“truth” ordering)

\[ [(M, s) \models AX \phi] = \begin{cases} 
\text{true} & \text{if } \forall s' : s \xrightarrow{\text{may}} s' \Rightarrow [(M, s') \models \phi] = \text{true} \\
\text{false} & \text{if } \exists s' : s \xrightarrow{\text{must}} s' \land [(M, s') \models \phi] = \text{false} \\
\bot & \text{otherwise} 
\end{cases} \]

- Ex: \([ (M, s) \models p ] = \text{true} \)
- Ex: \([ (M, s) \models AX p ] = \bot \)
Completeness Preorder

To measure the completeness of models (aka, refinement preorder, or abstraction\(^{-1}\).)

Let \(\preceq\) be the “information” ordering on truth values in which \(\bot \preceq \text{true}\) and \(\bot \preceq \text{false}\).

**Definition:** The completeness preorder \(\preceq\) is the greatest relation \(\preceq \subseteq S \times S\) such that \(s_a \preceq s_c\) implies the following:

- \(\forall p \in P : L_A(s_a, p) \leq L_C(s_c, p)\),
- if \(s_a \xrightarrow{\text{must}} A s'_a\), there is some \(s'_c \in S_C\) such that \(s_c \xrightarrow{\text{must}} C s'_c\) and \(s'_a \preceq s'_c\),
- if \(s_c \xrightarrow{\text{may}} C s'_c\), there is some \(s'_a \in S_A\) such that \(s_a \xrightarrow{\text{may}} A s'_a\) and \(s'_a \preceq s'_c\).

(Note: if no \(\bot\) and only \(\xrightarrow{\text{may}}\), \(\preceq\) is simulation.)

**Example:**

\[
\begin{array}{c}
\text{s}_a & \xrightarrow{\text{p=T}} & \xrightarrow{\text{p=F}} & \text{s}_c \\
\text{p=\bot} & \xrightarrow{} & \xrightarrow{} & \text{p=T}
\end{array}
\]
**Logical Characterization of Completeness Preorder**

**Theorem:** Let $\Phi$ denote the set of all formulas of 3-valued propositional modal logic. Then

$$s_a \preceq s_c \iff (\forall \phi \in \Phi : [s_a \models \phi] \leq [s_c \models \phi]).$$

Thus, models that are “more complete” with respect to $\preceq$ have more definite properties with respect to $\models$.

**Example:**

```
p=T
s_a ----> p=F

p=⊥
```

```
p=T
s_c

p=T
```

Page 5


**Completeness Preorder (Continued)**

**Corollary:**
Let $\Phi$ denote the set of all formulas of 3-valued propositional modal logic. Then

$$(\forall \phi \in \Phi : [(M_1, s_1) \models \phi] = [(M_2, s_2) \models \phi]) \text{ iff } (s_1 \preceq s_2 \text{ and } s_2 \preceq s_1).$$

**Note:** If $s_1$ and $s_2$ are bisimilar, this implies $s_1 \preceq s_2$ and $s_2 \preceq s_1$, but $s_1 \preceq s_2$ and $s_2 \preceq s_1$ does not imply $s_1$ and $s_2$ are bisimilar! [Bruns-G99]

**Example:** $s_0$ and $s'_0$ are not bisimilar, but cannot be distinguished by any formula of 3-valued propositional modal logic.
3-Valued Model Checking

**Problem:** Given a state \( s \) of a 3-valued model \( M \) and a formula \( \phi \), how to compute the value \([(M, s) \models \phi] \)?

**Theorem:** [Bruns-G00] The model-checking problem for a 3-valued temporal logic can be reduced to two model-checking problems for the corresponding 2-valued logic.

**STEP 1:** complete \( M \) into two “extreme” complete Kripke structures, called the **optimistic** and **pessimistic** completions:

- Extend \( P \) to \( P' \) such that, for every \( p \in P \) there exists a \( p' \in P' \) such that \( L(s, p) = \text{comp}(L(s, p')) \) for all \( s \) in \( S \).
- \( M_o = (S, L_o, \xrightarrow{\text{must}}) \) with
  \[
  L_o(s, p) \overset{\text{def}}{=} \begin{cases}
  \text{true} & \text{if } L(s, p) = \bot \\
  L(s, p) & \text{otherwise}
  \end{cases}
  \]
- \( M_p = (S, L_p, \xrightarrow{\text{may}}) \) with
  \[
  L_p(s, p) \overset{\text{def}}{=} \begin{cases}
  \text{false} & \text{if } L(s, p) = \bot \\
  L(s, p) & \text{otherwise}
  \end{cases}
  \]
3-Valued Model Checking (Continued)

STEP 2: transform $\phi$ to its positive form $T(\phi)$ with $T(\neg p) = \bar{p}$.

STEP 3: evaluate $T(\phi)$ on $M_o$ and $M_p$ using traditional 2-valued model checking, and combine the results:

$$[(M, s) \models \phi] = \begin{cases} 
true & \text{if } (M_p, s) \models T(\phi) \\
false & \text{if } (M_o, s) \not\models T(\phi) \\
\bot & \text{otherwise}
\end{cases}$$

This can be done using existing model-checking tools!

**Corollary:** 3-valued model checking has the same complexity as traditional 2-valued model checking.
Examples

Application:
Generation of a partial Kripke structure from a partial state-space exploration such that, by construction, $s'_0 \preceq s_0$ [Bruns-G99].

Examples:

- $[s_1 | A(true \cup p)] = true$
- $[s_2 | A(true \cup p)] = \bot$
- $[s_3 | A(true \cup p)] = false$
New 3-Valued Semantics

Observation: One can argue that the previous semantics returns \( \perp \) more often than it should...

Example: In a state \( s_a \) where \( p = \perp \) and \( q = \text{true} \),

\[
[s_a \models q \land (p \lor \neg p)] = \perp
\]

while the same formula is \text{true} in every complete state \( s_c \) such that \( s_a \preceq s_c \)!

New 3-valued “thorough” semantics: [Bruns-G00]

\[
[(M, s) \models \phi]_t = \begin{cases} 
\text{true} & \text{if } (M', s') \models \phi \text{ for all } (M', s') : s \preceq s' \\
\text{false} & \text{if } (M', s') \not\models \phi \text{ for all } (M', s') : s \preceq s' \\
\perp & \text{otherwise}
\end{cases}
\]

Is model checking more expensive with this semantics?

YES! Indeed, in general, one needs to solve two

Generalized Model-Checking Problems
Generalized Model Checking [Bruns-G00]

**Definition:** Given a state $s$ of a model $M$ and a formula $\phi$ of a temporal logic $L$, is there a state $s'$ of a complete system $M'$ such that $s \preceq s'$ and $(M', s') \models \phi$?

This **generalized model-checking problem** is thus a generalization of both **satisfiability** (all Kripke structures are potential solutions) and **model checking** (a single Kripke structure needs to be checked).

![Diagram of SAT and MC problems](image)

**Theorem:** The satisfiability problem for a temporal logic $L$ is reducible (in linear-time and logarithmic space) to the generalized model-checking problem for $L$.

Thus, GMC is as hard as satisfiability. Is it harder?
Branching-Time Temporal Logics

**Theorem:** (CTL) Given a state $s_0$ of partial Kripke structure $M = (S, L, \mathcal{R})$ and a CTL formula $\phi$, one can construct an alternating Büchi word automaton $A_{(M, s_0), \phi}$ over a 1-letter alphabet with at most $O(|S| \cdot 2^{|\phi|})$ states such that

$$(\exists (M', s'_0) : s_0 \preceq s'_0 \text{ and } (M', s'_0) \models \phi) \iff \mathcal{L}(A_{(M, s_0), \phi}) \neq \emptyset.$$ 

**Corollary:** if such a $M'$ exists, there exists one with at most $|S| \cdot 2^{|\phi|}$ states.

**Theorem:** The generalized model-checking problem for a state $s_0$ of a partial Kripke structure $M = (S, L, \mathcal{R})$ and a CTL formula $\phi$ can be decided in time $O(|S|^2 \cdot 2^{|\phi|}).$

**Theorem:** The generalized model-checking problem for CTL is EXPTIME-complete.

**Theorem:** (Summary) Let $L$ denote propositional logic, propositional modal logic, CTL, or any branching-time logic including CTL (such as CTL* or the modal $\mu$-calculus). The generalized model-checking problem for the logic $L$ has the same complexity as the satisfiability problem for $L.$
Linear-Time Temporal Logics

**Theorem:** (LTL) Given a state $s_0$ of partial Kripke structure $M = (S, L, R)$ and an LTL formula $\phi$, one can construct an alternating Büchi word automaton $A_{(M,s_0),\phi}$ over a 1-letter alphabet with at most $O(|S| \cdot 2^{1|\phi|})$ states such that

$$\exists (M', s'_0) : s_0 \leq s'_0 \text{ and } (M', s'_0) \models \phi \iff \mathcal{L}(A_{(M,s_0),\phi}) \neq \emptyset.$$

**Theorem:** The generalized model-checking problem for a state $s_0$ of a partial Kripke structure $M = (S, L, R)$ and an LTL formula $\phi$ can be decided in time $O(|S|^2 \cdot 2^{2|\phi|})$.

**Theorem:** The generalized model-checking problem for linear-time temporal logic is EXPTIME-complete.

For LTL, generalized model checking is thus **harder** than satisfiability and model checking! [Bruns-G00]

(both of these problems are PSPACE-complete for LTL)

Note: similar phenomenon for “realizability” and “synthesis” for LTL specifications [Abadi-Lamport-Wolper89, Pnueli-Rosner89].
Summary on Complexity in $|\phi|$ 

**Model Checking:** (3-valued semantics)
- MC can be reduced to two 2-valued MC problems.
- MC has the same complexity as 2-valued MC.

**Generalized Model Checking:** (thorough 3-val. sem.)
- For BTL, GMC has the same complexity as satisfiability.
- For LTL, GMC is harder than satisfiability and MC.

<table>
<thead>
<tr>
<th>Logic</th>
<th>MC</th>
<th>SAT</th>
<th>GMC</th>
</tr>
</thead>
<tbody>
<tr>
<td>PL</td>
<td>Linear</td>
<td>NP-Complete</td>
<td>NP-Complete</td>
</tr>
<tr>
<td>PML</td>
<td>Linear</td>
<td>PSPACE-Complete</td>
<td>PSPACE-Complete</td>
</tr>
<tr>
<td>CTL</td>
<td>Linear</td>
<td>EXPTIME-Complete</td>
<td>EXPTIME-Complete</td>
</tr>
<tr>
<td>$\mu$-calculus</td>
<td>NP∩co-NP</td>
<td>EXPTIME-Complete</td>
<td>EXPTIME-Complete</td>
</tr>
<tr>
<td>LTL</td>
<td>PSPACE-Complete</td>
<td>EXPTIME-Complete</td>
<td>EXPTIME-Complete</td>
</tr>
</tbody>
</table>
Complexity of GMC in $|M|$ 

Upper bound: can be done in quadratic time in $|M|$ [Bruns-G00].

**Theorem:** [G-Jagadeesan02] Checking emptiness of nondeterministic Büchi tree automata is reducible (in linear time and logarithmic space) to GMC for LTL (or CTL) properties represented by nondeterministic Büchi word (resp. tree) automata.

**Bad News:** (Lower bound) The best algorithm known for checking emptiness of nondeterministic Büchi tree automata $A$ requires quadratic time in $|A|$ in the worst case [Vardi-Wolper86].

**Good News:** better complexity for GMC and properties recognizable by nondeterministic co-Büchi word/tree automata, i.e., *persistence properties* (e.g., LTL formulas of the form $\Diamond \Box p$).

**Theorem:** [G-Jagadeesan02] GMC for persistence properties can be solved in time linear in $|M|$.

**Note:** persistence properties include all safety ($\Box p$) and guarantee ($\Diamond p$) properties. (Do not include $\Box \Diamond p$.)
Application: Automatic Abstraction

Idea: Given a concrete system $C$, if $C \models \phi$ cannot be decided, generate a (smaller) abstraction $A$ and check $A \models \phi$ instead.

Example: predicate abstraction

- Let $\psi_1, \ldots, \psi_n$ be $n$ predicates on variables of $C$.
- Abstract states are vectors of $n$ bits $b_i$.
- A concrete state $c$ is abstracted by an abstract state
  \[ [c] = (b_1, \ldots, b_n) \text{ iff } \forall 1 \leq i \leq n : b_i = \psi_i(c). \]

State of the art: $A$ is a traditional 2-valued model with
\[ (c_1 \rightarrow c_2) \Rightarrow ([c_1] \rightarrow [c_2]). \]
In other words, $A$ simulates $C$. Remember, this implies:

- If $\phi$ is a $\forall$-property, $A \models \phi$ implies $C \models \phi$,
- but $A \not\models \phi$ does not imply anything about $C$!
Automatic Abstraction Revisited

**Observation:** $A$ should really be a 3-valued model!

For instance, $A$ can be represented by a modal transition system.

**Abstraction relation:**

1. $(c_1 \rightarrow c_2) \Rightarrow ([c_1] \rightarrow_{\text{may}} [c_2])$
2. $(\forall c_i \in [a] : \exists c_i \rightarrow c_j \wedge c_j \in [a']) \Rightarrow ([a] \rightarrow_{\text{must}} [a'])$

By construction, $A \preceq C$.

Computing an MTS $A$ using (1)+(2) can be done at the same computational cost (same complexity) as computing a “conservative” abstraction (simulation) using (1) alone: (2) can be built by dualizing all the steps necessary to build (1).

This is shown for predicate and cartesian abstraction in [G-Huth-Jagadeesan01].
Automatic Abstraction Process

Traditional iterative abstraction procedure:

1. Abstract: compute $M_A$ that simulates $M_C$.
2. Check: given a universal property $\phi$, check $M_A |= \phi$.
   - if $M_A |= \phi$: stop (the property is proved: $M_C |= \phi$).
   - if $M_A \not|= \phi$: go to Step 3.
3. Refine: refine $M_A$. Then go to Step 1.

New procedure for automatic abstraction: (3 improvements)

1. Abstract: compute $M_A$ such that $M_A \preceq M_C$ (same cost as above [GHJ01])
2. Check: given any property $\phi$,
   1. (3-valued model checking) compute $[M_A |= \phi]$.
      - if $[M_A |= \phi] = \text{true}$ or $\text{false}$: stop.
      - if $[M_A |= \phi] = \bot$, continue.
   2. (generalized model checking) compute $[M_A |= \phi]_t$.
      - if $[M_A |= \phi]_t = \text{true}$ or $\text{false}$: stop.
      - if $[M_A |= \phi]_t = \bot$, go to Step 3.
3. Refine: refine $M_A$. Then go to Step 1.
Example

Predicate abstraction with $p$ : “is x odd?” and $q$ : “is y odd?” such that $M_2 \preceq C_2$:

program C2() {
    x,y = 1,0;
    x,y = 2*f(x),f(y);
    x,y = 1,0;
}

For $\phi_2 = \Diamond q \land \Box(p \lor \neg q)$, $[(M_2, s_2) \models \phi_2] = \bot$, but $[(M_2, s_2) \models \phi_2]_t = false$
(i.e., there does not exist a concretization of $(M_2, s_2)$ that satisfies $\phi_2$).

Thus, GMC is more precise than MC in this case.

(Same for the safety property $\phi'_2 = \lozenge q \land \Box(p \lor \neg q)$.)
Precision of GMC Vs. MC

How often is GMC more precise than MC? See [G-Huth05]:

- Studies when it is possible to reduce $\text{GMC}(M, \phi)$ to $\text{MC}(M, \phi')$.
- $\phi'$ is called a semantic minimization of $\phi$.
- Shows that PL (already known), PML, and $\mu$-calculus are closed under semantic minimization, but not LTL, CTL or CTL*.
- Identifies self-minimizing formulas, i.e., $\phi$’s for which $\text{GMC}(M, \phi) = \text{MC}(M, \phi)$
  - semantically (using automata-theoretic techniques, EXPTIME-hard in $|\phi|$ for $\mu$-calculus) and
  - syntactically (sufficient criterion only, linear in $|\phi|$).
- Ex (syntactic): Any formula that does not contain any atomic proposition in mixed polarity (in its negation normal form) is self-minimizing.
- Note: the converse is not true (e.g., $(\neg q_1 \lor q_2) \land (\neg q_2 \lor q_1)$ is self-minimizing).
- For any self-minimizing formula, GMC and MC have the same precision.
- Good news: many frequent formulas of practical interest are self-minimizing, and MC is as precise as GMC for those.
3-Valued Abstractions for Open Systems

**Open system:** system interacting with its environment.

**Module Checking (ModC)** [Kupferman-Vardi96]: given an open system $M$ and a formula $\phi$, does $M$ satisfy $\phi$ in *all possible environments*?

**Example:** (vending machine) is it always possible for $M$ to eventually serve tea?

- $MC(M, \text{ AGEF tea}) = true$
- $\text{ModC}(M, \text{ AGEF tea}) = false$

**Generalized Module Checking (GModC)** [G03]: given $A$ and $\phi$, does there exist a concretization $C$ of $A$ such that $C$ satisfies $\phi$ in all possible environments?

Two simultaneous games here: one with the environment, one with $\bot$ values...

Yet, GModC can be solved at the same cost as GMC (for LTL and BTL) [G03].
3-Valued Abstractions for Games

Study abstractions of games where moves of each player can now be abstracted, while preserving winning strategies of both players [de Alfaro-G-Jagadeesan04]:

- An abstraction of a game is now a game where each player has both may and must moves (yielding may/must strategies).
- Completeness preorder is now an alternating refinement relation, logically characterized by 3-valued alternating μ-calculus [Alur-Henzinger-Kupferman02].
- If must transitions are allowed to be nondeterministic [Larsen-Xinxin90], then the abstraction is as precise as can be, i.e., the framework is complete (see also [Namjoshi03, Dams-Namjoshi04]):
  “Given any infinite-state system C and property φ ∈ μ-calculus, if C satisfies φ, then there exists a finite-state abstraction A such that A satisfies φ.”

Example: [Namjoshi03]
- var x;
  actions (-) x:=x-1; (+) x:=x+1;
  property: EF(P) with P = (x ≥ 0)

- The construction of abstraction is now compositional (cf. [G-Huth-Jagadeesan01], [Shoham-Grumberg04], [de Alfaro-G-Jagadeesan04]).
Conclusions

3-Valued models and logics can be used to check any property, while guaranteeing soundness of counter-examples.

*Generalized Model Checking* means checking whether there exists a concretization of an abstraction that satisfies a temporal logic formula.

It can be used to improve precision of automatic abstraction, for a reasonable cost:

- Cost can be higher in the size of the formula...
  but only worst-case and formulas are short.
- Cost can be higher (quadratic) in the size of the model...
  but is the same (linear) for persistence properties (includes safety).

In an “abstract-check-refine” procedure, GMC is only polynomial in the size of the abstraction, and may prevent the unnecessary generation and analysis of possibly exponentially larger refinements of that abstraction.

In practice, use first a syntactic formula check for self-minimization: MC has then the same precision as GMC (often the case).
Other Related Work

“Mixed transition systems” [Dams-Gerth-Grumberg94]

- Intuitively, a mixed transition system is an MTS without the constraint $\underset{\text{must}}{\rightarrow} \subseteq \underset{\text{may}}{\rightarrow}$.
- Hence, more expressive than 3-valued models: some mixed TS cannot be refined into any complete system.
- Still, their goal is very similar (i.e., design may/must abstractions for MC).

“Extended transition systems” [Milner81]

- $\text{XTS} = \text{LTS} + \ “\text{divergence predicate”}$
- In [Bruns-G99], it is shown that 3-valued Hennessy-Milner Logic logically characterizes the “divergence preorder” [Milner81, Walker90].
- Close correspondence with Plotkin’s intuitionistic modal logic (inspired Bruns-G00 reduction from 3-val to 2-val MC).

3-Valued logic for program analysis: [Sagiv-Reps-Wilhelm99] shape graphs, first-order 3-valued logic, “focussing”, ... (roughly inspired the beginning of this work but technical details are fairly different e.g., no 3-valued abstraction on control)

Conservative abstraction for the full mu-calculus: [Saidi-Shankar99]