3-Valued Abstraction and 3-Valued Model-Checking

**Abstraction**

- **Abstraction:**
  - an effective technique to combat state explosion problem
  - approximate sets of concrete states by an abstract state
  - approximate sets of concrete transitions by an abstract transition

- **Using 2-valued logic (over-approximation)**
  - False variables represent “unknown” value
  - True transitions represent possible behaviour
    
    \[
    r(s_{0,1}) = T \Rightarrow r(s_0) = T \quad p(s_{0,1}) = F \Rightarrow ?
    \]
Abstraction, Cont’d

Using 2-valued logic

- False variables represent “unknown” value
- True transitions represent possible behaviour

\[ AX (r \lor \rho)(s_{0,1}) = \top \Rightarrow AX (r \lor \rho)(s_0) = \top \]
\[ AX r(s_{0,1}) = \bot \Rightarrow ? \]
\[ EX (\neg r \land \neg \rho)(s_{0,1}) = \bot \Rightarrow EX (\neg r \land \neg \rho)(s_0) = \bot \]
\[ EX \ \kappa(s_{0,1}) = \top \Rightarrow ? \]

Abstraction, Cont’d

Soundness:

- Only with respect to True universal properties
  - For existential properties – use under-approximation
- For False properties:
  - play counter-example to determine whether spurious
  - Use counter-example-based abstraction refinement
3-valued abstraction

**Goals:**
- Reason about mixed properties
- Not have to tell which counterexamples are spurious
- Not have an increase in statespace, when compared to 2-valued
- Use counterexample for abstraction refinement

**Outline:**
- 3-valued logic, properties, models, model-checking
- 3-valued abstractions
- Abstraction refinement

---

**Logic: 3-valued Kleene logic**

**Logic order**

**Properties:**
- $F \sqsubseteq M, M \sqsubseteq T$
- $A \land B = \min (A, B)$
- $A \lor B = \max (A, B)$
- $\neg T = F, \neg F = T, \neg M = M$

**Preserves:**
- Commutativity, associativity, idempotence, De Morgan laws

**Does not preserve**
- Law of excluded middle: $A \lor \neg A = T$ (top)
- Law of non-contradiction: $A \land \neg A = \bot$ (bottom)
Note

3-valued logic forms a lattice
- Ordering $\sqsubseteq$ : less than or equal
- Meet operation $\sqcap$ : min
- Join operation $\sqcup$ : max
- Negation : horizontal symmetry

This is an example of a quasi-boolean algebra

Equality and Identity are different!
- $a \not\equiv b$
- $a = b$

Logic

Information order

- $M$ contains least amount of information
- $T, F$ – maximum amount of information
- If one refines $M$ – it can change to $T$ or $F$ or stay at $M$
Multi-valued state machines: Xkripke structures

- Extension of conventional state machines (Kripke structures)
  - variables take any value from the logic \( (T, F, M) \)
  - transitions between states take any value from the logic
    - False transitions are not shown (by convention)

- Example:
Formally,

**Kripke structures extended for MV case**

\[ M = \langle L, S, A, s_0, I, R \rangle \]

- \( L \) is a quasi-boolean algebra \( (\mathcal{L}, \sqsubseteq, \sqcap, \sqcup, \neg) \), where \( (\mathcal{L}, \sqsubseteq) \) is a lattice
- \( S \) is a (finite) set of states, each with a unique name
- \( A \) is a set of atomic propositions
- \( s_0 \) is a unique initial state \( (s_0 \in S) \)
- \( I: S \times A \rightarrow \mathcal{L} \) is the interpretation function that assigns a logic value to each atomic proposition
- \( R: S \times S \rightarrow \mathcal{L} \) is the function that assigns a logic value to each transition between states

**3-valued CTL**

**multi-valued extension of CTL**

- same syntax as CTL
- plus constants from the logic \( (T, M, F) \)

**semantics:**

- replace existential quantification by disjunction, universal quantification by conjunction, so
  \[ (EX \phi)(s) = \lor_{t \in S} (SR(s,t) \land R(t)) \land \phi(t) \]
- other operators are defined as in CTL:

For all states \( s \),

\[ (AX \phi)(s) = (\neg EX(\neg \phi))(s) \]
\[ (EG \phi)(s) = \phi(s) \land (EX EG \phi)(s) \]
\[ (AG \phi)(s) = (\neg EF(\neg \phi))(s) \]

Examples:

- \( AG (request \rightarrow AX pressed) \not\models \)
- \( AG (pressed \lor request) \not\models \)
Model-Checking Cont’d

• Can a True property evaluate to M?

• Answer:
  - Yes
  - AG (pressed V ¬ pressed) = M
  - Comes from law of excluded middle

• Some terminology:
  - Compositional semantics
    ➢ Evaluate each CTL operator, compose according to lattice rules
  - Thorough semantics [Bruns&Godefroid 00]
    ➢ Property evaluates to M iff exists a refinement where it evaluates to T and a refinement where it evaluates to F.
    ➢ $$ to evaluate

Symbolic mv model-checking

• Similar idea to classical model-checking
  - recursively go through the structure of XCTL property
  - encode sets of states symbolically
  - encode transition relation symbolically

• Data structures
  - direct approach: MDDs
    ➢ the number of terminal nodes and branching factor equal to number of values in logic
    ➢ Example: x\&y in 3-valued logic
  - can use BDD vector …
  - or mixed approaches (MBTDDs, MTBDDs)
Reduction to Classical

[Bruns&Godefroid’99]. Assumption: transition relation is classical

- Move negation to level of atomic propositions
- Create a positive and negative version of every atomic proposition
- Let \( x = M \).
- Positive cut:
  - Set \( x \) and \( \neg x \) to True
  - PosAnswer = check property
- Negative cut:
  - Set \( x \) and \( \neg x \) to False
  - NegAnswer = check property

*If NegAnswer = PosAnswer (True or False)*

- Return this as answer

*Else*

- Return Maybe

---

Example

\[
[[p \land \neg q \lor z]](s_0) = T
\]
Example

\[ \text{Model} \]
\[ p - T \quad q - M \quad z = F \]
\[ p - T \quad q - F \quad z = M \]
\[ p - T \quad q - F \quad z = F \]

\[ \text{Negative Cut} \]
\[ p - T \quad q - F \quad z = F \]
\[ p - T \quad q - F \quad z = F \]
\[ p - T \quad q - F \quad z = F \]

\[ [[p \land \neg q \lor z]](s_0) = F \]
Therefore, the answer is \( M \)

Example

\[ \text{Model} \]
\[ p - T \quad q - M \quad z = F \]
\[ p - T \quad q - F \quad z = M \]
\[ p - T \quad q - F \quad z = F \]

\[ \text{Positive Cut} \]
\[ p - T \quad q - F \quad z = F \]
\[ p - T \quad q - F \quad z = F \]
\[ p - T \quad q - F \quad z = F \]

\[ [[\text{EX} (p \land \neg q \lor z)]](s_0) = T \]
Example

Model

\[ p \leftarrow T, q \leftarrow F, z \leftarrow F \]

\[ p \leftarrow T, q \leftarrow F, z \leftarrow M \]

Negative Cut

\[ p \leftarrow T, q \leftarrow F, z \leftarrow F \]

\[ p \leftarrow T, q \leftarrow F, z \leftarrow F \]

\[ [[\text{EX} (p \land \neg q \lor z)](s_0) = T \]

Therefore, the answer is \( T \)

---

Reduction to Classical (Take Two)

\[ \text{[Gurfinkel\&Chechik 2003]} \]

Assumptions:

- States can be 3-valued, transition relation can be three-valued

Reduction steps

- for True and Maybe, construct a cut formula equivalent to \([\phi](s) \equiv j\)
  - logic: from mv CTL to restricted mv-logic with two-valued answers
  - model: unchanged
- transform each cut to a classical model-checking problem
  - logic: from restricted mv-logic to classical CTL
  - model: from XKripke structure to classical Kripke structure
Propositional Logic

\[\begin{align*}
[[p \land \neg q \lor z]](s_0) &= M \\
[[T \land \neg M \lor F]](s_0) \\
[[T \land M \lor F]](s_0) \\
[[M \lor F]](s_0) \\
[[M]](s_0)
\end{align*}\]

Propositional Logic – the cut

\[\begin{align*}
[[p \land \neg q \lor z]](s_0) &\equiv T \\
\| p \land \neg q \lor z \| (s_0) &\equiv M \\
[[p \equiv T] \land (\neg q \equiv T) \lor (z \equiv T)](s_0) \\
[[T \land F \lor F]](s_0) \\
[[F \lor F]](s_0) \\
F &\equiv T
\end{align*}\]
Combining Results

\[ \lnot q \lor z ](s_0) \not\equiv T \]
\[ [p \land q \lor z ](s_0) \equiv M \]
Therefore, \( [p \land q \lor z ](s_0) = M \)

Propositional Logic – final step

\[ [[(p \equiv T) \land (\neg q \equiv T) \lor (z \equiv T)](s_0) \]
\[ [[p^+ \land q^+ \lor z^+]](s_0) \]
\[ [[T \land F \lor F]](s_0) \]
\[ [[F]](s_0) \]
Existential Temporal Logic – the cut

\[ [[\exists x (p \land \neg q \lor z)](s_0)] \equiv T \]
\[ \forall_{t \in S} R(s_0, t) \land [[p \land \neg q \lor z](t)] \equiv T \]
\[ \forall_{t \in S} (R(s_0, t) \equiv T) \land [[p \land \neg q \lor z](t)] \equiv T \]
\[ \forall_{t \in S} (R(s_0, t) \equiv T) \land [[(p \equiv T) \land (\neg q \equiv T) \lor (z \equiv T)](t)] \]
\[ [[\exists_{\equiv T} ((p \equiv T) \land (\neg q \equiv T) \lor (z \equiv T))] (s_0)] \]

Existential Temporal Logic – final step

\[ [[\exists_{\equiv T} ((p \equiv T) \land (\neg q \equiv T) \lor (z \equiv T))] (s_0)] \]
\[ [[\exists_{\equiv T} (p^+ \land q^- \lor z^-)] (s_0)] \]
\[ [[\exists (p^+ \land q^- \lor z^-)] (s_0)] \]
Universal Temporal Logic – the cut

Dealing with negation

- In 3-valued logic
  - $\neg b \equiv T$ iff $\neg (b \equiv M)$
  - since $\neg b \equiv T$ iff $b = F$

\[
[[AX(p \land \neg q \lor z)](s_0)](s_0) \equiv T
\]
\[
\land_{teS} R(s_0, t) \Rightarrow [[p \land \neg q \lor z]](t) \equiv T
\]
\[
\land_{teS} \neg R(s_0, t) \lor [[p \land \neg q \lor z]](t) \equiv T
\]
\[
\land_{teS} \neg R(s_0, t) \equiv T \lor [[p \land \neg q \lor z]](t) \equiv T
\]
\[
\land_{teS} \neg (R(s_0, t) \equiv M) \lor [[p \land \neg q \lor z]](t) \equiv T
\]
\[
\land_{teS}(R(s_0, t) \equiv M) \Rightarrow [[p \land \neg q \lor z]](t) \equiv T
\]
[[AX_{\equiv M} ((p \equiv T) \land (\neg q \equiv T) \lor (z \equiv T))]][s_0]

Universal Temporal Logic – final step

\[
[[AX_{\equiv M} ((p \equiv T) \land (\neg q \equiv T) \lor (z \equiv T))]][s_0]
\]
[[AX_{\equiv M} (p^+ \land q^+ \lor z^+)]][s_0]
[[AX(p^+ \land q^+ \lor z^+)]][s_0]
Handling Mixed Modalities

The first reduction step does not change

\[ [[AX \ EX \rho]](s_0) \models T \] is transformed into \[ [[AX_{\exists M} \ EX_{\exists T}(\rho \models T)]](s_0) \]

Problem with the second step

need a Kripke structure with two types of transitions

- \( \exists M \) for universal modality
- \( \exists T \) for existential modality

Solution

- treat transitions labels as actions
- convert the resulting Labeled Transition System into a Kripke structure

Disadvantage

- introduces a new variable
- size of the statespace doubles

Summary of the Reduction

Multi-valued model-checking problem is reduced to several classical problems

- one classical problem for True and one for Maybe
- size of the formula does not change
  - atomic literals are changed to “plus” and “minus” versions
  - other parts remain unchanged
- for universal and existential fragments
  - statespace of resulting Kripke structure is similar to the original
- for formulas with both universal and existential modalities
  - statespace of the resulting Kripke structure is double of the original
- formulas with fixpoint operators are handled similarly
  - (see Gurfinkel, Chechik, CONCUR’03)
Abstraction

Using 3-valued logic

- introduce new special value *Maybe* to stand for “unknown”

Formally:

- $[[\nu]](a) = T$ iff $\forall s \in \gamma(a)$ $[[\nu]](s) = T$
- $[[\nu]](a) = F$ iff $\forall s \in \gamma(a)$ $[[\nu]](s) = F$
- $[[\nu]](a) = M$ iff $\exists s \in \gamma(a)$ $[[\nu]](s) = T$ and $\exists t \in \gamma(a)$ $[[\nu]](t) = F$

Examples:

- $r(s_{0,1}) = T \Rightarrow r(s_0) = T$
- $\rho(s_{0,1}) = M$

Abstraction Function $\alpha : S \rightarrow S'$
Refresher: Over- and Under-approximations

- $M'$ is an over-approximation of $M$, or $M'$ simulates $M$ if
  \[ \exists R^{\exists} \text{[Dams'97]} : (t, t_1) \in R' \iff \exists s \in \gamma(t) \text{ s.t. } \exists s_1 \in \gamma(t_1) \text{ and } (s, s_1) \in R \]

- $M'$ is an under-approximation of $M$, or $M$ simulates $M'$ if
  \[ \forall R^{\forall} \text{[Dams'97]} : (t, t_1) \in R' \iff \forall s \in \gamma(t) \text{ s.t. } \exists s_1 \in \gamma(t') \text{ and } (s, s_1) \in R \]

Existential Abstraction (Over-Approximation)
Universal Abstraction (Under-Approximation)

3-Val Transition Relation

\[ \forall s, t \in R^{\exists} \text{ iff } \forall s \in \gamma(t) \text{ s.t. } \exists s_1 \in \gamma(t') \text{ and } (s, s_1) \in R \]

\[ \forall R^{\exists} \text{ [Dams'97]: } (t, t_1) \in R' \text{ iff } \exists s \in \gamma(t) \text{ s.t. } \exists s_1 \in \gamma(t_1) \text{ and } (s, s_1) \in R \]

\[ \text{Else } R(s,t) = M \]
**3-valued abstraction**

![Diagram of 3-valued abstraction](image)

**Abstraction**

*Using 3-valued logic*

introduce new special value *Maybe* to stand for “unknown”

\[ AX \ r(s_{0,1}) = M \]
\[ EX \ r(s_{0,1}) = T \]

![Diagram of abstraction](image)
Model Checking 3-Val abstract Models

- Let $\varphi$ be an arbitrary property (i.e., expressed in LTL, CTL, mu-calculus) and $M'$ is a 3-val abstraction of $M$

  - Preservation Theorem
    
    $$M' \models \varphi \iff M \models \varphi$$

- No guarantee is given about a “Maybe” answer
- False counterexample cannot be spurious
- No need for simulation!
- Maybe counterexample requires refinement

3-Val Abstraction-Refinement Loop

- $M, \varphi, \alpha$ → Abstract
- $M', \varphi$ → Model Check
- $\alpha'$ → Refine
- Spurious → Check Counterexample
- Pass → No Bug
- Fail → Bug

- $\varphi$
No spurious counterexamples, but abstraction can be too coarse

Refinement

Refinement : α'
**Other use of 3-valued logic**

- **Algebra:**
  - Use three-valued algebra (Kleene)
  - Intermediate value represents incomplete information or uncertainty
  - Compact representation for all possible refinements of this model
  - If a property is True/False on the partial model, it is True/False on a refined one
  - Initial theory developed by Bruns & Godefroid, CAV’99

**Application:**
- Most models are incomplete!
- Allows verification before specification is completed

---

**Summary**

- **Abstraction**
  - Effective tool for combating state explosion
  - Over-approximation – sound for true universal properties, otherwise – check if counterexample is feasible and then refine
  - Under-approximation – same for existential properties

- **3-Valued Abstraction**
  - Specified in 3-val Kleene logic
  - Allows reasoning about mixed-quantifier properties
  - No need to check if counter-example is spurious
  - Counterexample used for refinement

- **3-Val Model-Checking**
  - Reduces to two runs of classical model-checker
  - Or can be done directly, say, using MDDs
Next topic:

Software model-checking

(and software model-checking with 3-valued logic)