Program correctness
Model-checking CTL

Marcello Bonsangue
Formal Verification

Verification techniques comprise

- a modelling framework $M, \Gamma$
  to describe a system
- a specification language $\phi$
  to describe the properties to be verified
- a verification method $M \models \phi, \Gamma \vdash \phi$
  to establish whether a model satisfies a property
Model Checking

- **Question**: does a given transition system satisfy a temporal formula?
- **Simple answer**: use definition of $\models$!
  - We cannot implement it as we have to unwind the transition system in a possibly **infinite** tree.

Can we do better?

and most probably!
The problem

- We need **efficient** algorithms to solve the problems
  
  \[ [1] \quad M,s \models \phi \]
  \[ [2] \quad M,s \models \phi \]

  where $M$ should have finitely many states, and $\phi$ is a CTL formula.

- We concentrate to solution of [2], as [1] can be easily derived from it.
The solution

- **Input:** A CTL model M and CTL formula $\phi$
- **Output:** The set of states of M which satisfy $\phi$
- **Basic principles:**
  - Translate any CTL formula $\phi$ in terms of the connectives $\text{AF}$, $\text{EU}$, $\text{EX}$, $\land$, $\neg$, and $\perp$.
  - Label the states of M with sub-formulas of $\phi$ that are satisfied there, starting from the smallest sub-formulas and working outwards towards $\phi$.
  - Output the states labeled by $\phi$.
The labelling

- An immediate sub-formula of a formula $\phi$ is any maximal-length formula $\psi$ other than $\phi$ itself.
- Let $\psi$ be a sub-formula of $\phi$ and assume the states of $M$ have been already labeled by all immediate sub-formulas of $\psi$.
- Which states have to be labeled by $\psi$?

We proceed by case analysis.
The basic labeling

- \(\bot\) no states are labeled
- \(p\) label a state \(s\) with \(p\) if \(p \in l(s)\)
- \(\phi_1 \land \phi_2\) label a state \(s\) with \(\phi_1 \land \phi_2\) if \(s\) is already labeled with \(\phi_1\) and \(\phi_2\)
- \(\neg\phi\) label a state \(s\) with \(\neg\phi\) if \(s\) is not already labeled with \(\phi\)
The EX labeling

- $\text{EX}\phi$: Label with $\text{EX}\phi$ any state $s$ with one of its successors already labeled with $\phi$
The EU labeling

- \( E[\phi_1 U \phi_2] \equiv \phi_2 \lor (\phi_1 \land EXE[\phi_1 U \phi_2]) \)

1. Label with \( E[\phi_1 U \phi_2] \) any state \( s \) already labeled with \( \phi_2 \)
2. Repeat until no change: label any state \( s \) with \( E[\phi_1 U \phi_2] \) if \( s \) is labeled with \( \phi_1 \) and at least one of its successor is already labeled with \( E[\phi_1 U \phi_2] \)

- Diagram:
  - Initial state labeled with \( \phi_1 \)
  - Transition labeled with \( E[\phi_1 U \phi_2] \)
  - Repeat until no change
  - Final state labeled with \( E[\phi_1 U \phi_2] \)
The AF labeling

- \( AF\phi \equiv \phi \lor AXAF\phi \)

1. Label with \( AF\phi \) any state \( s \) already labeled with \( \phi \)
2. Repeat until no change: label any state \( s \) with \( AF\phi \) if all successors of \( s \) are already labeled with \( AF\phi \)

\[ \]

... until no change
The EG labeling (direct)

- \( \text{EG} \phi \equiv \phi \land \text{EXEG} \phi \equiv \neg \text{AF} \neg \phi \)

1. Label all the states with \( \text{EG} \phi \)
2. Delete the label \( \text{EG} \phi \) from any state \( s \) not labeled with \( \phi \)
3. Repeat until no change: delete the label \( \text{EG} \phi \) from any state \( s \) if none of its successors is labeled with \( \text{EG} \phi \)

\[ \text{repeat} \quad \ldots \text{until no change} \]
Complexity

The complexity of the model checking algorithm is \( O(f \cdot V \cdot (V+E)) \)

where

- \( f \) = number of connectives in \( \phi \)
- \( V \) = number of states of M
- \( E \) = number of transitions of M

It can be easily improved to an algorithm linear both in the size of the formula and of the model.
State explosion

- The algorithm is linear in the size of the model but the size of the model is **exponential** in the number of variables, components, etc.

Can we reduce state explosion?

- Abstraction (what is relevant?)
- Induction (for ‘similar’ components)
- Composition (divide and conquer)
- Reduction (prove semantic equivalence)
- Ordered binary decision diagrams
Example: Input

\[ \phi = AF(E[\neg q U p] v EXq) \]
Example: EU - step 1

1. Label with $E[\neg q \land p]$ all states which satisfy $p$
Example: EU-step 2.1

2.1 label with $E[\neg q]$ any state that is already labeled with $\neg q$ and with one of its successor already labeled by $E[\neg q]$
Example: EU-step 2.2

2.2 label with $E[\neg q]$ any state that is already labeled with $\neg q$ and with one of its successor already labeled by $E[\neg q]$
Example: EX-step 3

3. Label with EXq any state with one of its successors already labeled by q
Example: $\lor$-step 4

4. Label with $\sigma = E[\neg qUp] \lor EXq$ any state $s$ already labeled by $E[\neg qUp]$ or $EXq$
5.1 Label with $\phi = \text{AF}(E[\neg q\cup p] \lor \text{EX}q)$ any state already labeled by $\sigma = E[\neg q\cup p] \lor \text{EX}q$.
Example: AF-step 5.2

5.2 Label with $\phi$ any state with all successor already labeled by $\phi$. 
Example: Output

- All states satisfy $AF(E[\neg q U p] \lor EXq)$