MODEL CHECKING

- Exhaustive exploration of the state-space of a program.
  - If an error state is not reached, then model checking outputs safe.
  - If an error state is reached, then the path to the error state can be reconstructed, resulting in a counterexample.
- Model Checking for sequential programs comes in many variants:
  - Concrete Model Checking
  - Symbolic Model Checking
  - Bounded Model Checking
  - Abstract Model Checking

CONCRETE MODEL CHECKING

```
ConcreteModelChecking($\Gamma_c$,P)
  worklist := $\{(l_0,\sigma) \mid \sigma \in P\}$;
  reach := $\emptyset$;
  while worklist $\neq \emptyset$ do{
    Choose $(l,\sigma) \in$ worklist;
    worklist := worklist \ $\{(l,\sigma)\}$;
    if $((l,\sigma) \notin$ reach) then
    {
      reach := reach $\cup \{(l,\sigma)\}$;
      foreach $((l,c,l') \in T)$
         worklist := worklist $\cup \{(l',\sigma') \mid \sigma' \in sp(\{\sigma\},c)\}$;
    }
  }
  if $((l_{err},\_) \in$ reach) then
    return UNSAFE
  else
    return SAFE
```

CONCRETE MODEL CHECKING

WITH COUNTEREXAMPLE GENERATION

```
ConcreteModelChecking($\Gamma_c$,P)
  worklist := $\{(l_0,\sigma) \mid \sigma \in P\}$; parents := $\lambda x . NR$;
  reach := $\emptyset$;
  while worklist $\neq \emptyset$ do{
    Choose $(l,\sigma) \in$ worklist;
    worklist := worklist \ $\{(l,\sigma)\}$;
    if ($(l,\sigma) \notin$ reach) then
    {
      reach := reach $\cup \{(l,\sigma)\}$;
      foreach ($(l,c,l') \in T \wedge (l',\sigma') \in sp(\{\sigma\},c)$){
         worklist := worklist $\cup \{(l',\sigma')\}$;
         parents($(l',\sigma')$) := $(l,\sigma)$;
      }
    }
  }
  if ($(l_{err},\_) \in$ reach) then
    return UNSAFE
  else
    return SAFE
```

SYMBOLIC MODEL CHECKING

```
SymbolicModelChecking($\Gamma_c$,P)
  worklist := {($l_0$,$P$)};
  reach($l_0$) := $P$;
  foreach ($l \in L \backslash \{l_0\}$) reach($l$) := $false$;
  while worklist ≠ ∅ do{
    Choose $(l, F) \in$ worklist;
    worklist := worklist \ $\{(l, F)\}$;
    if (reach($l$) $\nRightarrow F$) then
    {
      reach($l$) := reach($l$) $\vee F$;
      foreach ($(l, c, l') \in T$)
         worklist := worklist $\cup \{(l', sp(F, c))\}$;
    }
  }
  if (reach($l_{err}$) $\neq false$) then
    return UNSAFE
  else
    return SAFE
```

BOUNDED MODEL CHECKING

- Concrete/Symbolic model checking for a finite number of steps
  - Unroll loops in the program for a fixed number of iterations, and then do concrete/symbolic model checking on the resultant program.
- Alternatively, we can apply Static Single Assignment (SSA) transformation on the unrolled program, and directly encode the BMC problem in FOL.

ABSTRACT MODEL CHECKING

- All the previous approaches to model checking have severe limitations:
  - Concrete and Symbolic Model Checking may not terminate and are in general computationally expensive.
  - Bounded Model Checking can only be used to find bugs, and not for verification.
- Let's bring back abstraction!
  - Consider a sound Abstract Interpretation framework $(D, \leq, \alpha, \gamma, \hat{F})$.

ABSTRACT MODEL CHECKING

```
AbstractModelChecking($\Gamma_c$,P)
  worklist := {($l_0$,$\alpha(P)$)};
  reach := $\emptyset$;
  while worklist $\neq$ $\emptyset$ do{
    Choose $(l,d) \in$ worklist;
    worklist := worklist \ $\{(l,d)\}$;
    if ( $\nexists (l,d') \in$ reach . $d \leq d'$) then
    {
      reach := reach $\cup \{(l,d)\}$;
      foreach ($(l,c,l') \in T$)
        worklist := worklist $\cup \{(l',d') \mid d' = \hat{f}_c(d)\}$;
    }
  }
  if ($(l_{err},d) \in$ reach $\wedge\, d \neq \bot$ ) then
    return UNSAFE
  else
    return SAFE
```

ABSTRACT MODEL CHECKING

WITH COUNTEREXAMPLE GENERATION

```
AbstractModelChecking($\Gamma_c$,P)
  worklist := {($l_0$,$\alpha(P)$)}; parents := $\lambda x . NR$;
  reach := $\emptyset$;
  while worklist $\neq$ $\emptyset$ do{
    Choose $(l,d) \in$ worklist;
    worklist := worklist \ $\{(l,d)\}$;
    if ( $\nexists (l,d') \in$ reach $. d \leq d'$) then
    {
      reach := reach $\cup \{(l,d)\}$;
      foreach ($(l,c,l') \in T$){
        worklist := worklist $\cup \{(l',\hat{f}_c(d))\}$;
        parents($(l',\hat{f}_c(d))$) := $(l,d)$;
      }
    }
  }
  if ($(l_{err},d) \in$ reach $\wedge d \neq \bot$ ) then
    return UNSAFE
  else
    return SAFE
```

PREDICATE ABSTRACTION

- The predicate abstraction domain is parameterized by a fixed, finite set of predicates P.
    - Each predicate is a formula over the program variables.
    - Example: $P = \{x \leq 1, y = 0, x + y \leq -1\}$
- There are two predicate abstraction domains:
    - Boolean Predicate Abstraction
    - Cartesian Predicate Abstraction

CARTESIAN PREDICATE ABSTRACTION

- The abstract domain is $\mathbb{P}(P) \cup \{ \perp \}$
- The partial order relation $\sqsubseteq$ is defined as follows:
  - $\forall s \in \mathbb{P}(P) \, . \perp \sqsubseteq s$
  - $\forall s_1, s_2 \in \mathbb{P}(P) \, . \, s_1 \sqsubseteq s_2 \Leftrightarrow s_1 \supseteq s_2$
- Top element is $\varnothing$, bottom element is $\perp$
- Example: $P = \{x \leq 1, y = 0, x + y \leq -1\}$. Which of the following are true?
  - $\{x \leq 1\} \sqsubseteq \{x \leq 1, x + y \leq -1\}$
  - $\{x + y \leq = 1, y = 0\} \sqsubseteq \{y = 0\}$
  - $\{x \leq 1\} \sqsubseteq \varnothing$

CARTESIAN PREDICATE ABSTRACTION

- Abstraction function:
  $\forall c \in \mathbb{P}(State) \,.\, c \neq \emptyset \Rightarrow \alpha(c) = \{p \in P \mid \forall \sigma \in c \,.\, \sigma \vDash p\}$
  - $\alpha(\emptyset) = \bot$
- Concretization function: $\forall s \in \mathbb{P}(P) \,.\, \gamma(s) = \{\sigma \mid \sigma \vDash \bigwedge_{p \in s} p\}$
  - $\gamma(\bot) = \emptyset$
- Examples $P = \{x \leq 1, y = 0, x + y \leq -1\}$
  - $\alpha(\{(0,0)\}) = \{x \leq 1, y = 0\}$
  - $\alpha(\{(0,0), (-1,-1)\}) = \{x \leq 1, x + y \leq -1\}$
  - $\alpha(x \leq 0) = \{x \leq 1\}$