Invariant Synthesis

Weikun Yang

July 31, 2019

1 Grammars

1.1 Current Implementation

$$\langle \texttt{Cand} \rangle ::= \langle \texttt{Ante} \rangle \implies \langle \texttt{Conseq} \rangle \tag{1.1}$$

$$\langle \texttt{Ante} \rangle ::= \langle \texttt{CSpred} \rangle \wedge \langle \texttt{Ante} \rangle \mid \mathit{true} \tag{1.2}$$

$$\langle \texttt{CSpred} \rangle ::= \langle \texttt{CSvar} \rangle = \langle \texttt{Const} \rangle \mid \langle \texttt{CSvar} \rangle \neq \langle \texttt{Const} \rangle \tag{1.3}$$

$$\langle \texttt{Conseq} \rangle ::= \langle \texttt{Dconj} \rangle \mid \langle \texttt{Dconj} \rangle \vee \langle \texttt{Conseq} \rangle \tag{1.4}$$

$$\langle \texttt{Dconj} \rangle ::= \langle \texttt{COpred} \rangle \wedge \langle \texttt{Dconj} \rangle \mid \langle \texttt{Dpred} \rangle \wedge \langle \texttt{Dconj} \rangle \mid \mathit{true} \tag{1.5}$$

$$\langle \texttt{COpred} \rangle ::= \langle \texttt{COvar} \rangle = \langle \texttt{Const} \rangle \mid \langle \texttt{COvar} \rangle \neq \langle \texttt{Const} \rangle \tag{1.6}$$

$$\langle \texttt{Dpred} \rangle ::= \langle \texttt{DOvar} \rangle = \langle \texttt{Const} \rangle \mid \langle \texttt{DOvar} \rangle \neq \langle \texttt{Const} \rangle \mid \langle \texttt{DOvar} \rangle = \langle \texttt{DIvar} \rangle \oplus \langle \texttt{DIvar} \rangle \tag{1.7}$$

Grammar parameters:

1. Variable categories:
   (a) `CSvar` control state
   (b) `COvar` control output
   (c) `DIvar` data input
   (d) `DOvar` data output
2. Size of `Ante`: number of predicates conjunction.
3. Bit-width of `Const`.
4. Size of `Conseq`: number of predicates in disjunction.
5. Size of `Dconj`: number of predicates in conjunction.
6. Supported bit-vector operations in `Dpred`: bv-add, bv-and, bv-neg etc.

1.2 Grammar with Variable Groups

$$\langle \texttt{Cand} \rangle ::= \langle \texttt{Ante} \rangle \implies \langle \texttt{Conseq} \rangle \tag{1.8}$$

$$\langle \texttt{Ante} \rangle ::= \langle \texttt{Cpred} \rangle \wedge \langle \texttt{Ante} \rangle \mid true \tag{1.9}$$

$$\langle \texttt{Cpred} \rangle ::= \langle \texttt{Cvar} \rangle = \langle \texttt{Const} \rangle \mid \langle \texttt{Cvar} \rangle \neq \langle \texttt{Const} \rangle \tag{1.10}$$

$$\langle \texttt{Conseq} \rangle ::= \langle \texttt{Group} \rangle \wedge \langle \texttt{Group} \rangle \mid \langle \texttt{Group} \rangle \vee \langle \texttt{Group} \rangle \tag{1.11}$$

$$\langle \texttt{Group} \rangle ::= \langle \texttt{Dpred} \rangle \wedge \langle \texttt{Dpred} \rangle \mid \langle \texttt{Dpred} \rangle \vee \langle \texttt{Dpred} \rangle \tag{1.12}$$

$$\langle \texttt{Dpred} \rangle ::= \langle \texttt{Dvar} \rangle = \langle \texttt{Const} \rangle \mid \langle \texttt{Dvar} \rangle \neq \langle \texttt{Const} \rangle \tag{1.13}$$

2 Pseudo-code for Term Enumeration

2.1 Current Implementation

Main function: `EnumerateCandidates`.

Algorithm 1: Enumerate Invariant Candidates

function `EnumerateCandidates`($CSvars$, $COvars$, $DIvars$, $DOvars$)**:**
  load $AnteSize, ConseqSize, DconjSize, BVOPs, MAXbw$ **global**
  $PredMap \leftarrow$ `EmptyMap`
  `EnumeratePredicates`($CSvars$, $MAXbw$)
  `EnumeratePredicates`($COvars$, $MAXbw$)
  `EnumeratePredicates`($DOvars$, $MAXbw$)
  `EnumerateOperations`($DOvars$, $BVOPs$, $DIvars$)
  $SetofAnte \leftarrow$ `EnumerateSelectK`($CSvars$, $AnteSize$)
  $DconjVars \leftarrow COvars \cup DOvars$
  $SetofDconj \leftarrow$ `EnumerateSelectK`($DconjVars$, $DconjSize$)
  $SetofConseq \leftarrow$ `EmptySet`
  for each $\{c_1, c_2, ..., c_{ConseqSize}\} \in 2^{SetofDconj}$ **do**
    $SetofConseq \leftarrow SetofConseq \cup \{\bigvee_{i=1}^{ConseqSize} c_i\}$
  end
  for each $(Ante, Conseq) \in SetofAnte \times SetofConseq$ **do**
    yield "$Ante \implies Conseq$"
  end
end

2.2 Variable Groups

Algorithm 2: HELPER FUNCTIONS

function `EnumeratePredicates`($Vars$, $MAXbw$)**:**
- **for each** $V \in Vars$ **do**
  - $PredSet \leftarrow$ `EmptySet`
  - $B \leftarrow$ `min`$(MAXbw,$ `getBW`$(V))$
  - **for each** C **in** $0, 1, ...2^B - 1$ **do**
    - $PredSet \leftarrow PredSet \cup \{V = C, V \neq C\}$
  - **end**
  - $PredMap[V] \leftarrow PredMap[V] \cup PredSet$
- **end**

end

function `EnumerateOperations`($Vars$, $Operators$, $Operands$)**:**
- **for each** V **in** $Vars$ **do**
  - $PredSet \leftarrow$ `EmptySet`
  - **for each** $OP \in Operators$ **do**
    - $N \leftarrow$ `arity`(OP)
    - **for each** $\{o_1, o_2, ..., o_N\} \in 2^{Operands}$ **do**
      - $PredSet \leftarrow PredSet \cup \{V = OP(o_1, o_2, ..., o_N)\}$
    - **end**
  - **end**
  - $PredMap[V] \leftarrow PredMap[V] \cup PredSet$
- **end**

end

function `EnumerateSelectK`($Vars$, K)**:**
- **for each** $\{V_1, V_2, ..., V_n\} \in 2^{Vars}$ *where* $n \leq K$ **do**
  - $XProdSet \leftarrow PredMap[V_1] \times PredMap[V_2] \times ... \times PredMap[V_n]$
  - **for each** $(pred_1, pred_2..., pred_n) \in XProdSet$ **do**
    - **yield** $\bigwedge_{i=1}^{n} pred_i$
  - **end**
- **end**

end

Algorithm 3: Enumerate Select (Grouped Variables)

```
function EnumerateCandidates(Cvars, Dvars, Groups):
    load AnteSize = 1, MAXbw global PredMap ← EmptyMap
    EnumeratePredicates(Cvars, MAXbw)
    EnumeratePredicates(Dvars, MAXbw)
    SetofAnte ← EnumerateSelectK(Cvars, AnteSize)
    GroupPreds ← EnumerateSelectGroups(Groups, "CONJ")
    GroupPreds ← EnumerateSelectGroups(Groups, "DISJ")
    SetofConseq ← EmptySet
    for each {c_1, c_2} ∈ 2^GroupPreds do
        SetofConseq ← SetofConseq ∪ {c_1 ∧ c_2, c_1 ∨ c_2}
    end
    for each (Ante, Conseq) ∈ SetofAnte × SetofConseq do
        yield "Ante ⟹ Conseq"
    end
end
function EnumerateSelectGroups(Groups, Connective):
    for each {V_1, V_2, ..., V_n} ∈ Groups do
        XProdSet ← PredMap[V_1] × PredMap[V_2] × ... × PredMap[V_n]
        for each (pred_1, pred_2..., pred_n) ∈ XProdSet do
            if Connective = "CONJ" then
                yield ⋀_{i=1}^{n} pred_i
            end
            else if Connective = "DISJ" then
                yield ⋁_{i=1}^{n} pred_i
            end
        end
    end
end
```