Proving Conditional Termination

Maria Gorinova, University of Cambridge
T2 Temporal Logic Prover

- [https://github.com/mmjb/T2](https://github.com/mmjb/T2)
- T2 can:
  - Prove termination / nontermination.
  - Prove CTL properties (recently CTL* properties too).
  - Prove safety / reachability.
Why is conditional termination useful?

- Large programs: Hard to handle!
- ... maybe we can look at smaller parts?

- Example

\[
k = -1 \\
\ldots \\
\textbf{while} \ (x > 0) \ \textbf{do} \\
\quad x = x + k
\]
Why is conditional termination useful?

- Split the program into strongly connected components.
- Find preconditions for termination one by one, to reason about termination of the entire program.
So how can we find the inputs for which a program terminates?
Initial Condition and Transition Relation

- **Initial Condition:**
  - $\theta \equiv (x = 10)$

- **Transition Relation for Loop:**
  - $\rho(\{x, y, z\}, \{x', y', z'\}) \equiv (x > 0) \land (x' = x + y) \land (y' = y + z) \land (z' = z)$

---

```
x = 10
while (x > 0) do
    x = x + y
    y = y + z
```
Ranking Function

Ranking function: $rf$, s.t:

- it is bounded: $\forall x \in X. \; rf(x) \geq 0$
- it is strictly decreasing: $\forall x \in X. \; rf(x) \geq rf(x') + 1$

while ($x > 0$) do
  $x = x - 1$

Ranking Function: $rf(x) = x - 1$
while (x > 0) do
  x = x + y
  y = y + z
Potential Ranking Function: $\text{prf}$, s.t:

- **it is bounded:** $\forall \vec{x} \in X \times Y \times Z. \text{prf}(\vec{x}) \geq 0$

- Obtain by: $QELIM(\exists x', y'. \rho(x, y, x', y'))$

  $\equiv QELIM(\exists x', y'. (x > 0) \land (x' = x + y) \land (y' = y + z))$

  $\equiv x > 0$

**Potential** Ranking Function: $\text{prf}(\vec{x}) = x - 1$
From a PRF to a RF

**Strengthening:**

- A formula $s$, s.t. $\forall \bar{x} \in X \times Y \times Z. s(\bar{x}) \land \rho(\bar{x}, \bar{x}') \rightarrow \text{prf}(\bar{x}) \geq \text{prf}(\bar{x}') + 1$

**Obtain by:** $QELIM(\forall x', y'. \rho(x, y, x', y') \rightarrow \text{prf}(x, y) \geq \text{prf}(x', y') + 1)$
  
  $\equiv QELIM(\forall x', y'. (x > 0) \land (x' = x + y) \land (y' = y + z) \rightarrow (x' - 1 \leq x - 2))$

  $\equiv x \leq 0 \lor y < 0$

**Strengthening**: $s(\bar{x}) \equiv x \leq 0 \lor y < 0$

```plaintext
While (x > 0) do
  x = x + y
  y = y + z
```
From a PRF to a RF

Precondition for termination:

- A formula $p$, s.t $s(\vec{x})$ is invariant on each iteration

  - $\rho(\vec{x}, \vec{x}') = (x > 0) \land (x' = x + y) \land (y' = y + z)$
  - $\text{prf}(\vec{x}) = x - 1$
  - $s(\vec{x}) \equiv x \leq 0 \lor y < 0$

Preconditions:

$$p \equiv (x \leq 0) \lor (x + y \leq 0) \lor (y < 0 \land x + 2y + z \leq 0) \lor \cdots \lor (y < 0 \land z \leq 0)$$

while $(x > 0)$ do

$x = x + y$

$y = y + z$
Final Preconditions?

- Preconditions: \( p \equiv (x \leq 0) \)
  \( \lor (x + y \leq 0) \)
  \( \lor (y < 0 \land x + 2y + z \leq 0) \)
  \( \lor \ldots \)
  \( \lor (y < 0 \land z \leq 0) \)

- \( z \leq -1 \)
While (x > 0) do
    x = x + y
    y = y + z

Preconditions: 
\[ p \equiv (x \leq 0) \lor (x + y \leq 0) \lor (y < 0 \land x + 2y + z \leq 0) \lor \ldots \lor (y < 0 \land z \leq 0) \]

While (x > 0) do
    x = x + y
    y = y + z

while (x > 0 && y < 0 && z <= 0) do
    x = x + y
    y = y + z

while (x > 0 && !(y < 0 && z <= 0)) do
    x = x + y
    y = y + z

Constrain the transition relation with \( \neg (y < 0 \land z \leq 0) \)
Proceed recursively
Does that really work?
Behind the curtains...

- Finding the right invariant conditions
  - How do we get \( y < 0 \land z \leq 0 \)?

- Choosing the right constraints
  - \( p \equiv (x \leq 0) \lor (x + y \leq 0) \lor (y < 0 \land x + 2y + z \leq 0) \lor \cdots \lor (y < 0 \land z \leq 0) \)
  - Recurse with \( \text{while } (x > 0 \land !(x \leq 0)) \) ???
  - Or with \( \text{while } (x > 0 \land !(x + y \leq 0)) \) ???
  - ...

Behind the curtains…

- Filtering, filtering, filtering
- If-then-else
If – then – else

Outside of loops:

```plaintext
if(\(x > 0\)) \text{ then}
    \text{while}(y > 0) \text{ do}
    \quad y = y - x
\text{else}
    \text{while}(y > 0) \text{ do}
    \quad y = y + x

[(\(x > 0\)) \land \text{preSynthPhase}(P)] \lor
[(\(x \leq 0\)) \land \text{preSynthPhase}(Q)]
```
If – then – else

Inside a loop:

```plaintext
while (x < 10) do
  if (x > 0) then
    x = x + y
```

Diagram:

```
1 → 2 → 4 → 1
```

- x <= 0
- x > 0
- x < 10
- x = x+y
If – then – else

Inside a loop:

- **Left hand side**
  - Conditions for branch change: \( b_1 \equiv x > 0 \)
  - Conditions for proper exit: \( c_1 \equiv false \)

- **Right hand side**
  - Conditions for branch change: \( b_2 \equiv x \leq 0 \lor y < 0 \)
  - Conditions for proper exit: \( c_2 \equiv x \geq 10 \lor y > 0 \)

- **Final conditions:**
  \[
  (c_1 \lor (b_1 \land c_2)) \land (c_2 \lor (b_2 \land c_1)) \\
  \equiv (x \geq 10 \lor y > 0)
  \]
What happens next?

- Spilt the program into strongly connected components
- Start from the “end” ones
- Reason backwards
fun preSynth (rho)

  prf = findPRF (rho)
  s   = findStrengthening (rho, prf)
  cond = findCondForInvariance (rho, s)

return cond

fun preSynthPhase (rho)

  result      = preSynth (rho)
  constraints = getConstraints (result)

  for constr in constraints do
    result = result V preSynthPhase (rho ∧ ¬constr)
  end

return result