NUS
School of Computing

A Specification Logic for Termination and Non-Termination Reasoning

TON-CHANH LE

JOINT WORK WITH C. GHERGHINA, A. HOBOR AND W-N. CHIN

Motivation

```
int fact (int n)
 requires n≥0
 variance n
 ensures res=n!;
{ if (n == 0) return 1;
  else return n * fact(n-1); }
```

```
fact(-1);


int i = user_input();
fact(i);
```

Motivation

int fact (int n)
 requires $n<0 \wedge$ Loop $\vee\ n\geq 0 \wedge$ Term[n]
 ensures res=n!;

Temporal Constraints

- Definite Termination: Term M
- Definite Non-termination: Loop
- Possible Non-termination: MayLoop

Termination Logic Model

- Execution capacity $\mathcal{I} = (\mathcal{I}^l, \mathcal{I}^u)$ is a pair of bounds in $\mathbb{N} \cup \{\infty\}$ tracking the *minimum* and *maximum* length of execution allowed.
- Model for the temporal constraints ϑ

$$(\mathcal{S}, \mathcal{H}, \mathcal{I}) \vDash \vartheta \text{ iff } \mathcal{I} = \mathcal{L}(\vartheta)$$

  - $\mathcal{L}(Term\ M) = \left(0_f, \mathcal{O}(M)\right)$
  - $\mathcal{L}(Loop) = \left(\infty_f, \infty_f\right)$
  - $\mathcal{L}(MayLoop) = (0_f, \infty_f)$

Termination Logic Model

- Model for the temporal constraints ϑ
  - $(\mathcal{S}, \mathcal{H}, (\infty, \infty)) \vDash Loop$
  - $\big(\mathcal{S}, \mathcal{H}, (0, c)\big) \vDash Term\ M$, where $(\mathcal{S}, \mathcal{H}) \vDash \mathcal{O}(M) = c$
  - $(\mathcal{S}, \mathcal{H}, (0, \infty)) \vDash MayLoop$
- Model for formulae with temporal constraints

$$
\begin{array}{lcl}
(\mathcal{S}, \mathcal{H}, \mathcal{I}) \models \Psi_1 \vee \Psi_2 & \equiv & (\mathcal{S}, \mathcal{H}, \mathcal{I}) \models \Psi_1 \text{ or } (\mathcal{S}, \mathcal{H}, \mathcal{I}) \models \Psi_2 \\
(\mathcal{S}, \mathcal{H}, \mathcal{I}) \models \Psi_1 \wedge \Psi_2 & \equiv & (\mathcal{S}, \mathcal{H}, \mathcal{I}) \models \Psi_1 \text{ and } (\mathcal{S}, \mathcal{H}, \mathcal{I}) \models \Psi_2 \\
(\mathcal{S}, \mathcal{H}, \mathcal{I}) \models \exists x_i^* \cdot \Psi & \equiv & \exists \nu_i^* \cdot (\mathcal{S}[(x_i \mapsto \nu_i)^*], \mathcal{H}, \mathcal{I}) \models \Psi \\
(\mathcal{S}, \mathcal{H}, \mathcal{I}) \models \rho & \equiv & (\mathcal{S}, \mathcal{H}) \vDash \rho \\
(\mathcal{S}, \mathcal{H}, \mathcal{I}) \models \vartheta & \equiv & (\mathcal{S}, \mathcal{H}) \vDash \mathcal{L}(\vartheta) = \mathcal{I}
\end{array}
$$

Temporal Entailment

- Execution Splitting Operator $\rhd$: $\vartheta_1 \rhd \vartheta_2$ holds for any state from which the program can take as many steps as ϑ_1 prescribes and end in a state satisfying ϑ_2.
- Temporal Entailment $\vartheta \vdash \vartheta_1 \rhd \vartheta_2$

$$\texttt{MayLoop} \vdash \texttt{MayLoop} \triangleright \texttt{MayLoop}$$
$$\texttt{Loop} \vdash \texttt{MayLoop} \triangleright \texttt{Loop}$$
$$\texttt{MayLoop} \vdash \texttt{Loop} \triangleright \texttt{MayLoop}$$
$$\texttt{Loop} \vdash \texttt{Loop} \triangleright \texttt{MayLoop}$$

$$\frac{\texttt{MayLoop} \vdash \texttt{Term X} \triangleright \texttt{MayLoop} \qquad \texttt{Loop} \vdash \texttt{Term X} \triangleright \texttt{Loop} \qquad \rho \Longrightarrow \texttt{Y} <_d \texttt{X}}{\rho \wedge \texttt{Term X} \vdash \texttt{Term Y} \triangleright \texttt{Term X} -_d \texttt{Y}}$$

Logical Entailment

- Integrating the temporal entailment into the entailment system $\psi \vdash \phi \rightsquigarrow \phi_r$ of underlying logics

$$\frac{\rho_a \vdash \rho_c \rightsquigarrow \Phi_r \qquad \rho_a \wedge \Phi_r \wedge \vartheta_a \vdash \vartheta_c \triangleright \vartheta_r}{\rho_a \wedge \vartheta_a \vdash \rho_c \wedge \vartheta_c \rightsquigarrow (\Phi_r \wedge \vartheta_r)}$$

Example

data node { int val; node next; }

pred ll(self, n) = self=null $\wedge$ n=0
  $\vee$ $\exists$r. self $\mapsto$ node(_, r) * ll(r, n-1)

```
int length (node x)
requires ll(x, n) ∧ Term[n]
ensures res=n;
{
    if (x == null) return 0;
    else return 1 + length(x.next);
}
```

Example

$ll(x, n) \wedge x \neq null \wedge Term[n] \vdash x \mapsto node(\_, y) * ll(y, m) \wedge Term[m]$

$ll(x, n) \wedge x \neq null \vdash x \mapsto node(\_, y) * ll(y, m) \rightsquigarrow n = m+1$

$n = m+1 \wedge Term[n] \vdash Term[m] \rhd Term[1]$

Summary

- A logical framework for specifying and proving asserts about program termination and non-termination
- Soundness proofs for both termination and non-termination reasoning with an operational semantics instrumented by the execution capacity
- An integration of termination and non-termination reasoning into a separation logic based verification system. The implementation and the detailed paper are available at

http://www.comp.nus.edu.sg/~chanhle/