Temporal Reasoning for Machine Code

Nadeem Abdul Hamid

Berry College
Mount Berry, Georgia, U.S.A.
nadeem@acm.org

November 27, 2008
15th Int'l Conference on Logic for Programming, Artificial Intelligence and Reasoning

Context: Proof-Carrying Code

Proof-Carrying Code (PCC)

- Executable code packaged with a formalized proof of safety
- Selling points
  - Code consumer is provided static proof object (origin of code/proof is irrelevant)
  - Proof applies directly to binary machine code

Proof Properties

- Current PCC research focuses only on *safety* properties
  - Safety property: proposition that a certain state always or never holds
- This work
  - Consider how to certify more general correctness properties (liveness, deadlock-freeness, fairness in temporal logic terminology)

Logics

- Coq proof assistant
  - Higher-order predicate logic with inductive definitions
  - Used to encode alll definitions and proofs
- Temporal logic
  - Formalism for properties involving a notion of time

Approach

- Encode machine “syntax” and semantics in Coq (standard)
- Encode temporal logic operators in Coq (derive “inference rules”)
- Mechanism for building an “abstract automaton” based on a specific program
- Develop rules for reasoning about global properties (invariants) and eventuality properties (termination)

Sample: Counter Program

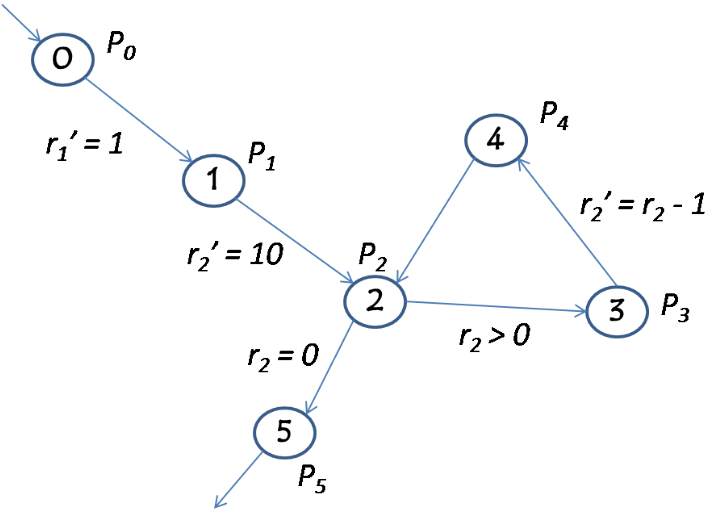
```
0     f0: movi r1 1
1         movi r2 10
2     f1: bz r2 f2
3         dec r2
4         goto f1
5     f2: nop
6         movi r1 0
...       ...
```

Program Trace

- (program counter, registers)

$(0, \{0{,}0{,}\ldots\})$
$\rightsquigarrow (1, \{1{,}0{,}\ldots\})$
$\rightsquigarrow (2, \{1{,}10{,}\ldots\})$
$\rightsquigarrow (3, \{1{,}10{,}\ldots\})$
$\rightsquigarrow (4, \{1{,}9{,}\ldots\})$
$\rightsquigarrow (2, \{1{,}9{,}\ldots\})$
$\rightsquigarrow (3, \{1{,}9{,}\ldots\})$
$\rightsquigarrow (4, \{1{,}8{,}\ldots\})$
$\rightsquigarrow (2, \{1{,}8{,}\ldots\})$
$\rightsquigarrow \ldots$

Automaton Abstraction



Related Work

- Proof-carrying code [Necula 1997; Appel et al. 2001; Hamid et al. 2003; Yu et al. 2004]
- Temporal logic and PCC [Bernard/Lee 2002; Henzinger 2002]
- Temporal logic in Coq [Coupet-Grimal 2003; Tsai/Wang 2008]
- Termination . . .
- Temporal proof generation . . .

Future Work

- Add nondeterminism in machine model (hardware interrupts)
- Address scalability/level of automation
  - Proof-generating model checker, or something similar
  - Derive annotations for program points from high-level source code

Thank you!

nadeem@acm.org

Prototype development in Coq:
`http://cs.berry.edu/~nhamid/pubs/minipic.coq.tgz`

Temporal Operators

$$(\mathcal{N}\ G)\ s_0 \triangleq \forall s_1.\ s_0 \leadsto s_1 \Rightarrow G\ s_0\ s_1$$

(G holds on every state that steps from s_0.)

$$(\mathcal{F}\ G)\ s_0 \triangleq \forall \overrightarrow{s_0}.\ \exists s_i \in \overrightarrow{s_0}.\ G\ s_0\ s_i$$

(Along every path from s_0, there is eventually a state s_i such that $G\ s_0\ s_i$.)

$$(\mathcal{A}\ G)\ s_0 \triangleq \forall \overrightarrow{s_0}.\ \forall s_i \in \overrightarrow{s_0}.\ G\ s_0\ s_i$$

(Every state s_i along every path from s_0 satisfies $G\ s_0\ s_i$.)

$$(G\ \mathcal{U}\ H)\ s_0 \triangleq \forall \overrightarrow{s_0}.\ \exists s_i \in \overrightarrow{s_0}.\ H\ s_0\ s_i\ \wedge\ \forall j < i.\ \forall s_j \in \overrightarrow{s_0}.\ G\ s_0\ s_j$$

(Along every path from s_0, there is eventually a state s_i such that $H\ s_0\ s_i$, and for every state s_j preceding s_i in the path, $G\ s_0\ s_j$ holds.)

Temporal Operator Rules

$$\frac{G\ s_0\ s_0 \qquad (\mathrm{step} \circ G)\ s_0 \ \Rightarrow\ G\ s_0}{(\mathcal{A}\ G)\ s_0}\ (\textsc{Always-Later})$$

$$\frac{G\ s_0\ s_0}{(\mathcal{F}\ G)\ s_0}\ \textsc{Evntly-Now} \qquad \frac{\begin{array}{c}\forall s_1 .\ s_0 \leadsto s_1 \ \Rightarrow\ (\mathcal{F}\ G')\ s_1 \\ (G' \circ \mathrm{step})\ s_0 \ \Rightarrow\ G\ s_0\end{array}}{(\mathcal{F}\ G)\ s_0}\ \textsc{Evntly-Later}$$

Notation

$$P \ \Rightarrow\ Q \ \triangleq\ \forall s.\ P\ s \ \Rightarrow\ Q\ s \qquad (P \ \Rightarrow\ Q)\ s \ \triangleq\ P\ s \ \Rightarrow\ Q\ s$$

$$(P \ \wedge\ Q)\ s \ \triangleq\ P\ s \ \wedge\ Q\ s \qquad (P \ \vee\ Q)\ s \ \triangleq\ P\ s \ \vee\ Q\ s$$

$$\overline{P}\ s\ s' \ \triangleq\ P\ s' \quad (\textit{lift state predicate to action})$$

$$G \ \Rightarrow\ H \ \triangleq\ \forall s, s'.\ G\ s\ s' \ \Rightarrow\ H\ s\ s'$$

$$(G \circ H)\ s\ s' \ \triangleq\ \exists s''.\ H\ s\ s'' \ \wedge\ G\ s''\ s'$$

$$(G \wedge H)\ s\ s' \ \triangleq\ H\ s\ s' \ \wedge\ G\ s\ s'$$

Abstract Automata

Abstract Automaton Components

$$
\begin{array}{rl}
(\textit{abstract state},\ t) & \text{abstate} : \text{Set} \\
(\textit{abstraction invariant},\ \text{inv}\ t\ s) & \text{inv} : \text{abstate} \to \text{state} \to \text{Prop} \\
(\textit{step abstraction},\ t \overset{G}{\rightsquigarrow} t') & \text{abstep} : \text{abstate} \to \text{abstate} \to \text{saction} \to \text{Prop}
\end{array}
$$

Simulation Properties

$$
\begin{array}{rl}
\text{invstep\_inv} : & \forall t, s, s'.\ (\text{inv}\ t\ s \ \wedge\ s \rightsquigarrow s') \ \Rightarrow\ \exists t', G.\ (t \overset{G}{\rightsquigarrow} t' \ \wedge\ \text{inv}\ t'\ s') \\
\text{abstep\_sane} : & \forall t, t', G.\ t \overset{G}{\rightsquigarrow} t' \ \Rightarrow\ \exists s, s'.\ \text{inv}\ t\ s \ \wedge\ \text{inv}\ t'\ s' \ \wedge\ s \rightsquigarrow s' \\
\text{abstep\_inv} : & \forall s, s', t, t', G.\ \text{inv}\ t\ s \ \wedge\ \text{inv}\ t'\ s' \ \wedge\ s \rightsquigarrow s' \ \wedge\ t \overset{G}{\rightsquigarrow} t' \ \Rightarrow\ G\ s\ s'
\end{array}
$$

$\Box$ and $\Diamond$ Rules

$$(\textit{labeling environment}) \quad \Gamma \;::=\; \cdot \mid \Gamma, (t : G)$$

$$\frac{\forall s.\ \mathrm{inv}\ t\ s \;\Rightarrow\; G\ s\ s}{\Gamma \vdash (\Diamond G)\ t}\,\Diamond\text{NOW} \qquad \frac{\begin{array}{c}\forall t', H.\ t \overset{H}{\rightsquigarrow} t' \;\Rightarrow\; \Gamma \vdash (\Diamond G')\ t' \\ (G' \circ H) \;\Rightarrow\; G\end{array}}{\Gamma \vdash (\Diamond G)\ t}\,\Diamond\text{LATER}$$

$$\frac{\begin{array}{c}(t : G') \in \Gamma \\ G' \;\Rightarrow\; G\end{array}}{\Gamma \vdash (\Box G)\ t}\,\Box\text{ENV} \qquad \frac{\begin{array}{c}\forall s.\ \mathrm{inv}\ t\ s \;\Rightarrow\; G\ s\ s \\ \forall t', H.\ t \overset{H}{\rightsquigarrow} t' \;\Rightarrow\; \Gamma, (t : G) \vdash (\Box G')\ t' \\ (G' \circ H) \;\Rightarrow\; G\end{array}}{\Gamma \vdash (\Box G)\ t}\,\Box\text{STEP}$$

Theorems

For all G, t, and s:

$$\frac{\cdot \vdash (\Diamond G)\ t \qquad \text{inv}\ t\ s}{(\mathcal{F}\ G)\ s}\ \Diamond\text{-SOUND} \qquad \frac{\cdot \vdash (\Box G)\ t \qquad \text{inv}\ t\ s}{(\mathcal{A}\ G)\ s}\ \Box\text{-SOUND}$$

The latter depends on a lemma, for all G, t, Γ, s, s', and n:

$$\frac{\Gamma \vdash (\Box G)\ t \qquad \text{inv}\ t\ s \qquad s \leadsto^{n} s' \qquad \text{wf\_env}\ \Gamma}{G\ s\ s'}\ \Box\text{\_INV}$$

Termination

$$\frac{\forall s.\ \text{lpinv}\ s_0\ s\ \Rightarrow\ (P\ \vee\ (\mathcal{F}\ (\text{lpinv}\ \wedge\ H)))\ s}{(\mathcal{F}\ (\text{lpinv}\ \wedge\ \overline{P}))\ s_0}\ \mathcal{F}\text{-TERM}$$

$$\frac{\forall s.\ \text{lpinv}\ s_0\ s\ \Rightarrow\ (P\ \vee\ (G\,\mathcal{U}\,(G\ \wedge\ \text{lpinv}\ \wedge\ H)))\ s}{(G\,\mathcal{U}\ (\text{lpinv}\ \wedge\ \overline{P}))\ s_0}\ \ (\mathcal{U}\text{-TERM})$$

where,

P : spred
H : saction, is a well-founded relation
lpinv : saction, is reflexive and transitive, and
G : saction, is a transitive relation

Idealized Processor Model

State Components

$$
\begin{aligned}
\text{(addresses, words)}\ pc, f, w, k &:= 0, 1, 2, 3, \ldots \\
\text{(register file)}\ \mathbb{R} &:= \{\mathrm{r}_0 \mapsto w_0, \mathrm{r}_1 \mapsto w_1, \ldots, \mathrm{r}_n \mapsto w_n\} \\
\text{(commands)}\ c &:= \text{movi } \mathrm{r}_i\ w \mid \text{dec } \mathrm{r}_i \mid \text{bz } \mathrm{r}_i\ f \mid \text{goto } f \mid \text{nop} \mid \ \ldots \ \mid \text{i} \\
\text{(code memory)}\ \mathbb{C} &:= \{0 \mapsto c_0, 1 \mapsto c_1, 2 \mapsto c_2, \ldots\} \\
\text{(state)}\ \mathbb{S} &:= (\mathbb{C}, \mathbb{R}, pc, k)
\end{aligned}
$$

Step Relation

$(\mathbb{C}, \mathbb{R}, pc, k) \ \rightsquigarrow \ \mathbb{S}'$	
if $\mathbb{C}(pc) =$	and $\mathbb{S}' =$
movi r_i w	$(\mathbb{C}, \mathbb{R}\{\mathrm{r}_i \mapsto w\}, pc + 1, k + 1)$
dec r_i	$(\mathbb{C}, \mathbb{R}\{\mathrm{r}_i \mapsto (\mathbb{R}(\mathrm{r}_i) - 1)\}, pc + 1, k + 1)$
bz r_i f	$(\mathbb{C}, \mathbb{R}, f, k + 1)$ if $\mathbb{R}(\mathrm{r}_i) = 0$
bz r_i f	$(\mathbb{C}, \mathbb{R}, pc + 1, k + 1)$ if $\mathbb{R}(\mathrm{r}_i) > 0$
goto f	$(\mathbb{C}, \mathbb{R}, f, k + 1)$
nop	$(\mathbb{C}, \mathbb{R}, pc + 1, k + 1)$
illegal	$(\mathbb{C}, \mathbb{R}, pc, k + 1)$

Program and Abstraction Invariant

(To reduce notational clutter, r_i and r'_i represent $\mathbb{R}(r_i)$ and $\mathbb{R}'(r_i)$, respectively.)

f	$\mathbb{C}_0(f)$	inv $f\ (\mathbb{C}, \mathbb{R}, pc, k) \triangleq (pc = f \land \mathbb{C} = \mathbb{C}_0 \land P)$, where P is
0	`f0: movi r1 1`	$r_1 = 0 \land r_2 = 0$
1	`movi r2 10`	$r_1 = 1 \land r_2 = 0$
2	`f1: bz r2 f2`	$r_1 = 1 \land r_2 \leq 10$
3	`dec r2`	$r_1 = 1 \land r_2 \leq 10 \land r_2 > 0$
4	`goto f1`	$r_1 = 1 \land r_2 \leq 10$
5	`f2: nop`	$r_1 = 1 \land r_2 = 0$
6	`movi r1 0`	$r_1 = 1 \land r_2 = 0$
7	`movi r2 10`	$r_1 = 0 \land r_2 = 0$
8	`f3: bz r2 f4`	$r_1 = 0 \land r_2 \leq 10$
9	`dec r2`	$r_1 = 0 \land r_2 \leq 10 \land r_2 > 0$
10	`goto f3`	$r_1 = 0 \land r_2 \leq 10$
11	`f4: goto f0`	$r_1 = 0 \land r_2 = 0$
≥ 12	`ill`	False

Example (I)

Step Abstraction

$f_0 \overset{G}{\leadsto} f_1$, where		
f_0	f_1	$G\ (\mathbb{C}, \mathbb{R}, pc, k)\ (\mathbb{C}', \mathbb{R}', pc', k')$
0	1	$k' = k + 1 \ \wedge\ \mathrm{r}_1' = 1 \ \wedge\ \mathrm{r}_i' = \mathrm{r}_i,\ i \neq 1$
1	2	$k' = k + 1 \ \wedge\ \mathrm{r}_2' = 10 \ \wedge\ \mathrm{r}_i' = \mathrm{r}_i,\ i \neq 2$
2	3	$k' = k + 1 \ \wedge\ \mathrm{r}_2 > 0 \ \wedge\ \mathrm{r}_i' = \mathrm{r}_i$
2	5	$k' = k + 1 \ \wedge\ \mathrm{r}_2 = 0 \ \wedge\ \mathrm{r}_i' = \mathrm{r}_i$
3	4	$k' = k + 1 \ \wedge\ \mathrm{r}_2' = \mathrm{r}_2 - 1 \ \wedge\ \mathrm{r}_i' = \mathrm{r}_i,\ i \neq 2$
4	2	$k' = k + 1 \ \wedge\ \mathrm{r}_i' = \mathrm{r}_i$
5	6	$k' = k + 1 \ \wedge\ \mathrm{r}_i' = \mathrm{r}_i$
...	...	...

Sample Property

$(\mathcal{N}\ (\mathcal{F}\ (\mathrm{r}_1' = 0\ \wedge\ k' = 34 + k)))\ \mathbb{S}_0$.
That is, from every next state of $\mathbb{S}_0$, a state is eventually reached where the value of $\mathbb{R}'(\mathrm{r}_1)$ is 0 and the clock is incremented by 34 cycles.

Applying the $\mathcal{F}$-TERM rule:
$P(\mathbb{C}, \mathbb{R}, pc, k)\ \triangleq\ \mathbb{R}(\mathrm{r}_2) = 0$
$H(\mathbb{C}, \mathbb{R}, pc, k)\ (\mathbb{C}', \mathbb{R}', pc', k')\ \triangleq\ \mathbb{R}'(\mathrm{r}_2) = \mathbb{R}(\mathrm{r}_2) - 1$
$\mathrm{lpinv}(\mathbb{C}, \mathbb{R}, pc, k)\ (\mathbb{C}', \mathbb{R}', pc', k')$
$\triangleq\ \mathbb{C}' = \mathbb{C} \wedge pc' = pc \wedge \mathbb{R}'(\mathrm{r}_1) = \mathbb{R}(\mathrm{r}_1) \wedge k' = k + 3 \times (\mathbb{R}(\mathrm{r}_2) - \mathbb{R}'(\mathrm{r}_2))$
H holds between successive iterations of the loop and is well-founded. lpinv relates the initial state at the beginning of the loop to the state at the top of the loop in every future iteration.