All-Termination(SCP)

Aaron Turon
Northeastern University
turon@ccs.neu.edu

(joint work with Pete Manolios)
“Why, sometimes I've believed as many as six impossible things before breakfast.”

-- Queen of Hearts, Alice in Wonderland
“Why, sometimes I've believed as many as six impossible things before breakfast.”

-- Queen of Hearts, Alice in Wonderland

**Slogan 0**

*All-Termination:*
how to solve six impossible problems before lunch
(define (insert i x xs)
  (cond
   ((<= i 0) (cons x xs))
   ((empty? xs) (list x))
   (else (cons (car xs)
                  (insert (- i 1) x
                           (cdr xs))))))
(define \textbf{insert} \textit{i} \textit{x} \textit{xs})
  (cond
    ((\leq \textit{i} \textbf{0}) (\textbf{cons} \textit{x} \textit{xs}))
    ((\textbf{empty?} \textit{xs}) (\textbf{list} \textit{x}))
    (else (\textbf{cons} (\textbf{car} \textit{xs})
             (\textbf{insert} (- \textit{i} 1)
            \textit{x}
            (\textbf{cdr} \textit{xs}))))

How do we prove that \textbf{insert} terminates?
\[(\text{define} \ (\text{insert} \ i \ x \ xs) \quad \text{(cond)} \quad (\leq i \ 0) \ (\text{cons} \ x \ xs)) \quad (\text{empty?} \ xs) \ (\text{list} \ x) \quad (\text{else}) \quad (\text{cons} \ (\text{car} \ xs) \quad (\text{insert} \ (- \ i \ 1) \ x \quad (\text{cdr} \ xs))))\]

How do we prove that \text{insert} terminates?

\[m_1(i, x, xs) = i\]
(define (insert i x xs)
  (cond
    ((<= i 0) (cons x xs))
    ((empty? xs) (list x))
    (else (cons (car xs)
                (insert (- i 1)
                        x
                        (cdr xs))))))

How do we prove that `insert` terminates?

\[ m_1(i, x, xs) = i \]
\[ m_2(i, x, xs) = \text{length } xs \]
(define (insert i x xs)
 (cond
   ((<= i 0) (cons x xs))
   ((empty? xs) (list x))
   (else (cons (car xs) (insert (- i 1) x (cdr xs))))))

How do we prove that insert terminates?

\[ m_1(i, x, xs) = i \]
\[ m_2(i, x, xs) = \text{length } xs \]
\[ m_3(i, x, xs) = i + \text{length } xs \]
(define (insert i x xs)
  (cond
   ((<= i 0) (cons x xs))
   ((empty? xs) (list x))
   (else (cons (car xs) (insert (- i 1) x (cdr xs))))))

How do we prove that \textit{insert} terminates?

\begin{align*}
m_1(i, x, xs) &= i \\
m_2(i, x, xs) &= \text{length } xs \\
m_3(i, x, xs) &= i + \text{length } xs
\end{align*}
(define (insert i x xs)
  (cond
   ((<= i 0)   (cons x xs))
   ((empty? xs) (list x))
   (else        (cons (car xs)
                 (insert (- i 1)
                  x
                    (cdr xs))))))

How do we prove that insert terminates?

Measured sets

for insert

{i}

{x}

{i, xs}
Measured sets $\implies$ induction schemes [Boyer&Moore, 1979]

To prove $\forall i, x, xs :: \varphi(i, x, xs)$, show:
Measured sets $\Rightarrow$ induction schemes [Boyer&Moore, 1979]

To prove $\forall i, x, xs :: \varphi(i,x,xs)$, show:

Measured set \{i\}

$\varphi(0,x,xs)$

$[\forall y, ys :: \varphi(i,y,ys)] \Rightarrow \varphi(i+1,x,xs)$
Measured sets $\Rightarrow$ induction schemes \cite{BoyerMoore1979}

To prove $\forall i,x,\text{xs} :: \varphi(i,x,\text{xs})$, show:

Measured set \{i\}

$\varphi(0,x,\text{xs})$

$[\forall y,\text{ys} :: \varphi(i,y,\text{ys})] \Rightarrow \varphi(i+1,x,\text{xs})$

Measured set \{\text{xs}\}

$\varphi(i,x,\text{nil})$

$[\forall j,y :: \varphi(j,y,\text{xs})] \Rightarrow \varphi(i,x,\text{cons}(z,\text{xs}))$
Measured sets $\implies$ induction schemes [Boyer&Moore, 1979]

To prove $\forall i,x,xs :: \varphi(i,x,xs)$, show:

Measured set $\{i\}$
- $\varphi(0,x,xs)$
- $[\forall y,ys :: \varphi(i,y,ys)] \implies \varphi(i+1,x,xs)$

Measured set $\{xs\}$
- $\varphi(i,x,nil)$
- $[\forall j,y :: \varphi(j,y,xs)] \implies \varphi(i,x,\text{cons}(z,xs))$

Measured set $\{i,xs\}$
- $\varphi(0,x,nil)$
- $[\forall y :: \varphi(i,y,xs)] \implies \varphi(i+1,x,\text{cons}(z,xs))$
Life with a theorem prover:

User

Define functions

Make conjectures

Known functions

Rewriting

Induction

Proving engine

Known facts

Theorem prover
Life with a theorem prover:

User

Theorem prover

- Define functions
- Make conjectures
- Prove termination

Known functions

- Rewriting
- Induction

Proving engine

Known facts
Life with **ACL2:**

- Define functions
- Make conjectures
- Prove termination

**Theorem prover**

- Known functions
- Rewriting
- Induction
- Proving engine
- Known facts

**User**
Life with **ACL2 Sedan:**

- Define functions
- Make conjectures
- Termination analysis
- Known functions
  - Rewriting
  - Induction
- Proving engine
  - Known facts
- Theorem prover

User
A better life ACL2 Sedan:

- Define functions
- Make conjectures

All-Termination analysis

- Known functions
  - Rewriting
  - Induction

Proving engine

- Known facts

Theorem prover

User
Slogan 1

Termination is not a yes/no question – it's multiple choice
The rest of the talk:

All-Termination($T$)

definition

research program

Poly-time size-change termination (SCP)

All-Termination(SCP)
Termination analysis

Termination undecidable

Sound, incomplete analyses:

\[ T : \text{Programs} \rightarrow \text{Bool predicate such that if } T(P) \text{ then } P \text{ terminates on all inputs} \]
Termination analysis

Termination undecidable

Sound, incomplete analyses:

\[ T : \text{Programs} \rightarrow \text{Bool} \text{ predicate such that} \]
\[ \text{if } T(P) \text{ then } P \text{ terminates on all inputs} \]

Restricted termination analysis

\[ T : \text{Programs} \times 2^{\text{Variables}} \rightarrow \text{Bool} \]

such that

\[ \text{if } T(P,V) \text{ then } V \text{ is a measured set for } P \]
Measured sets are upward-closed:

if $U \subseteq V$ and $U$ is a measured set for $P$ then so is $V$
Measured sets are upward-closed:

if $U \subseteq V$ and $U$ is a measured set for $P$ then so is $V$

All-Termination($T$) analysis

$$\text{All-Termination}(T)(P) \overset{\text{def}}{=} \text{minimal}\{V \mid T(P, V)\}$$

where $T$ is a restricted termination analysis.
Measured sets are upward-closed:

if $U \subseteq V$ and $U$ is a measured set for $P$ then so is $V$

All-Termination($T$) analysis

$$\text{All-Termination}(T)(P) \overset{\text{def}}{=} \text{minimal}\{V \mid T(P,V)\}$$

where $T$ is a restricted termination analysis.

Warning

$|\text{All-Termination}(T)(P)|$ can be exponential in $|P|$. 

Theorem:

if $T$ is in PSPACE then $\text{AllTermination}(T)$ is in PSPACE.

Proof:

$\text{All-Termination}(T)(P)$:

for each $V \subseteq \text{vars}(P)$

if $T(P,V)$ then

minimal := true

for each $U \not\subseteq V$

if $T(P,U)$ then minimal := false

if minimal then output($V$)
Research program

Begin with standard termination analysis, $A$

Define restricted version, $T$, so that

$$\exists V :: T(P,V) \iff A(P)$$

*Instrument* $A$ to produce a “certificate”

Implement All-Termination($T$)($P$) by

running $A$ on $P$ to produce certificate

*extracting* measured sets from certificate
Slogan 2

All-Termination does not increase power –
it enriches results
The rest of the talk:

All-Termination\((T)\)

Poly-time size-change termination (SCP)

All-Termination(SCP)
Size-change termination [Lee et al, POPL01] works by analyzing a safe abstraction of the program.

\[
\begin{align*}
\text{ack}(0,n) & = n+1 \\
\text{ack}(m,0) & = \text{ack}(m-1, 1) \\
\text{ack}(m,n) & = 2\text{ack}(m-1, 3\text{ack}(m, n-1))
\end{align*}
\]
Size-change termination [Lee et al, POPL01] works by analyzing a safe abstraction of the program.

\[
\begin{align*}
    \text{ack}(0,n) &= n+1 \\
    \text{ack}(m,0) &= \text{ack}(m-1, 1) \\
    \text{ack}(m,n) &= \text{ack}(m-1, \text{ack}(m, n-1))
\end{align*}
\]
Size-change termination [Lee et al, POPL01] works by analyzing a safe abstraction of the program.

\[
\begin{align*}
    \text{ack}(0, n) &= n+1 \\
    \text{ack}(m, 0) &= \text{ack}(m-1, 1) \\
    \text{ack}(m, n) &= 2\text{ack}(m-1, 3\text{ack}(m, n-1))
\end{align*}
\]
Size-change termination [Lee et al, POPL01] works by analyzing a safe abstraction of the program.

\[
\begin{align*}
\textbf{ack}(0, n) &= n+1 \\
\textbf{ack}(m, 0) &= \text{ack}(m-1, 1) \\
\textbf{ack}(m, n) &= \text{ack}(m-1, \text{ack}(m, n-1))
\end{align*}
\]

![Diagram of the ack function](image)
Size-change termination [Lee et al, POPL'01] works by analyzing a safe abstraction of the program.

\[
\begin{align*}
\text{ack}(0,n) &= n+1 \\
\text{ack}(m,0) &= \text{ack}(m-1,1) \\
\text{ack}(m,n) &= \text{ack}(m-1, \text{ack}(m,n-1))
\end{align*}
\]
Call sequence
Call sequence
Call sequence
Call sequence
**Thread**

\[
\begin{array}{cccccccc}
  m & \rightarrow & m & \rightarrow & m & \rightarrow & m & \rightarrow & m \\
  n & \rightarrow & n & \rightarrow & n & \rightarrow & n & \rightarrow & n \\
  3 & 2 & 2 & 2 & 3 & 3 & 3 & 3 & 3
\end{array}
\]

...
Program $P$ is **size-change terminating** for graph $G$ if:
each infinite path in $G$ has a thread w/ infinite descent

**Theorem** [Lee et al, POPL01]
Deciding size-change termination is $PSPACE$-complete
Poly-time size-change analysis

Call site C is an anchor iff:

Passing through C infinitely often entails infinite descent

Algorithm [Ben-Amram, Lee 2007]:

SCP(G):

\[
\begin{align*}
\text{for each } H & \text{ in } \text{SCC}(G) \\
A & := \text{FindAnchors}(H) \\
\text{if empty}(A) & \text{ or SCP}(H-A) = \text{false} \\
\text{then return } & \text{false} \\
\text{return } & \text{true}
\end{align*}
\]
The rest of the talk:

All-Termination($T$)

Poly-time size-change termination (SCP)

All-Termination(SCP)
A (naïve!) restricted version of SCP

Let \(\text{restrict}(G, V)\) be \(G\), but with only size-change edges relating variables in \(V\).

**Theorem:** if

1. \(G\) is a valid annotated call graph for \(P\)
2. \(\text{SCP}(\text{restrict}(G, V))\)

then \(V\) is a measured subset for \(P\).
Good, but ...
Good, but ...

**Theorem:**
Deciding $\exists V :: \text{SCP}(\text{restrict}(G,V))$ is NP-hard.
Good, but ...

**Theorem:**
Deciding $\exists V :: \text{SCP}(\text{restrict}(G, V))$ is NP-hard.

**Corollary:**
It is *not* true that $\text{SCP}(G)$ iff $\exists V :: \text{SCP}(\text{restrict}(G, V))$.

*What's going on?*
Good, but ...

**Theorem:**
Deciding $\exists V :: SCP(\text{restrict}(G,V))$ is NP-hard.

**Corollary:**
It is *not* true that $SCP(G)$ iff $\exists V :: SCP(\text{restrict}(G,V))$.

What's going on?

**Non-monotonicity:**
$V \subseteq W$ and $SCP(\text{restrict}(G,V))$ does **not** imply $SCP(\text{restrict}(G,W))$
Slogan 3

Nonmonotonicity means trouble
What we do

- Instrument SCP to produce an anchor tree
- Anchor tree is a certificate of termination
- Transform anchor tree to boolean constraint system $\varphi$
  - $\varphi$ captures which variables are required for the termination proof
    - small thread preservers are allowed!
- $|\varphi| = \mathcal{O}( |G| )$
Finding the minimal solutions

Constraints $\varphi$ are dual-horn: can find $\psi$ that is equisatisfiable to $\varphi$

- conjunction of clauses,
- each clause a disjunction of literals
- at most one negative literal per clause

min solutions to $\varphi$ can be found from $\psi$ efficiently
Theorem:

After computing $\varphi$, we can find $k$ elements of $\text{All-Termination}(\text{SCP})(P)$ in time $O(|G|^k)$

*Pay-as-you-go algorithm*
Slogan 4

To win, instrument and extract
Preliminary experimental results

ACL2 has a large regression suite:

>100MB

>11,000 function definitions (each of which must be proved terminating)

Code ranging from bit-vector libraries to model checkers
Preliminary experimental results

We have implemented, for ACL2,

- Poly-time size-change (SCP)
- Exp-time size-change (SCT)
- All-Termination (SCT)

We have *not* yet implemented

- All-Termination (SCP)
Preliminary experimental results

The setup: we ran CCG + All-Termination(SCT) on the entire regression suite.

Number of functions: >11,000
Proved terminating: 98% (note: same as CCG+SCT)

Multiargument functions:
  Proved terminating 1728
  With “nontrivial” cores 90%
  With multiple cores 7%
  Maximum core count 3

Running time (not including CCG): 30 seconds
Preliminary experimental results

The setup: we ran CCG + All-Termination(SCT) on the entire regression suite.

Number of functions: >11,000
Proved terminating: 98% (note: same as CCG+SCT)

Multiargument functions:
- Proved terminating: 1728
- With “nontrivial” cores: 90%
- With multiple cores: 7%
- Maximum core count: 3

Running time (not including CCG): 30 seconds
Preliminary experimental results

The setup: we ran CCG + All-Termination(SCT) on the entire regression suite.

Number of functions: >11,000
Proved terminating: 98% (note: same as CCG+SCT)

Multiargument functions:

- Proved terminating: 1728
- With “nontrivial” cores: 90%
- With multiple cores: 7%
- Maximum core count: 3 (the k parameter)

Running time (not including CCG): 30 seconds
Future work

Implement **All-Termination**(SCP)

Extend our prototype to the ACL2 Sedan

  Will help our freshman users at Northeastern

Study **All-Termination**(T) for additional T

  e.g. dependency-pair termination analysis

Explore new applications of measured subsets

  We've got a few in mind, but want to hear yours
Contribution recap

- Proposed the All-Termination($T$) problem
- Studied All-Termination(SCP)

Slogan recap

- Termination is not a yes/no question
- All-termination increases richness, not power
- Nonmonotonicity means trouble
- To win, instrument and extract