Trace Analysis for Predicting the Effectiveness of Partial Evaluation

Germán Vidal

Technical University of Valencia, Spain

International Conference on Logic Programming, **ICLP 2008**

Dec 8-13, 2008

Udine, Italy

Goal: analyze/predict the effects of **partial evaluation** of logic programs

First try:

- compare the original and the specialized programs

source program

```
...
add(int(X), int(Y), int(Z))  :- Z is X + Y.
add(str(X), str(Y), str(Z))  :- atom_concat(X, Y, Z).
add(int(X), str(Y), str(Z))  :- atom_number(X, SX), atom_concat(SX, Y, Z).
add(str(X), int(Y), str(Z))  :- atom_number(Y, SY), atom_concat(X, SY, Z).
...
inter([R], [R]).
inter([X, Y|T], R)  :- add(X, Y, Z), inter([Z|T], R).
```

source program

```
...
add(int(X), int(Y), int(Z)) :- Z is X + Y.
```

specialized program

```
...
inter__16(A, B, C, D, E) :- inter([str(A), int(B), str(C)|D], E).
inter__17(A, B, C, D) :- inter([str(A), str(B)|C], D).
inter__24(A, B, C, D, E) :- inter([str(A), str(B), str(C)|D], E).
inter__25(A, B, C, D, E, F, G, H) :- inter([str(A), B, str(C), int(D), str(E), str(F), B|G], H).
inter__36(A, B, C, D, E, F, G, H) :- inter([str(A), B, C, int(D), str(E), str(F), B, C|G], H).
...
```

- hard to analyze (only experimental approaches so far)
- no intuitition

source program

```
...
add(int(X), int(Y), int(Z)) :- Z is X + Y.
```

specialized program

```
...
inter__16(A, B, C, D, E) :- inter([str(A), int(B), str(C)|D], E).
inter__17(A, B, C, D) :- inter([str(A), str(B)|C], D).
inter__24(A, B, C, D, E) :- inter([str(A), str(B), str(C)|D], E).
inter__25(A, B, C, D, E, F, G, H) :- inter([str(A), B, str(C), int(D), str(E), str(F), B|G], H).
inter__36(A, B, C, D, E, F, G, H) :- inter([str(A), B, C, int(D), str(E), str(F), B, C|G], H).
...
```

- hard to analyze (only experimental approaches so far)
- no intuitition

source program

```
...
add(int(X), int(Y), int(Z)) :- Z is X + Y.
```

specialized program

```
...
inter__16(A, B, C, D, E) :- inter([str(A), int(B), str(C)|D], E).
inter__17(A, B, C, D) :- inter([str(A), str(B)|C], D).
inter__24(A, B, C, D, E) :- inter([str(A), str(B), str(C)|D], E).
inter__25(A, B, C, D, E, F, G, H) :- inter([str(A), B, str(C), int(D), str(E), str(F), B|G], H).
inter__36(A, B, C, D, E, F, G, H) :- inter([str(A), B, C, int(D), str(E), str(F), B, C|G], H).
...
```

- hard to analyze (only experimental approaches so far)
- no intuitition

Goal: analyze/predict the effects of **partial evaluation** of logic programs

Second try:

- ~~compare the original and the specialized program~~
- approximate the call traces of the original program with a CFG and transform it using the BTA annotations (prediction)

source program

$$
\begin{array}{l}
mlist([\,], I, [\,]).\\
mlist([X|R], I, L) \leftarrow ml(X, R, I, L).\\
ml(X, R, I, [XI|RI]) \leftarrow mult(X, I, XI),\ \ mlist(R, I, RI).\\
mult(0, Y, 0). \quad\ mult(s(X), Y, Z) \leftarrow mult(X, Y, Z1),\ \ add(Z1, Y, Z).\\
add(X, 0, X). \quad\ add(X, s(Y), s(Z)) \leftarrow add(X, Y, Z).
\end{array}
$$

source program

$$\begin{array}{l}
mlist([\,], I, [\,]).\\
mlist([X|R], I, L) \leftarrow ml(X, R, I, L).\\
ml(X, R, I, [XI|RI]) \leftarrow mult(X, I, XI),\ mlist(R, I, RI).
\end{array}$$

trace CFG

$$\begin{array}{lcl@{\qquad}rcl@{\qquad}rcl}
\mathrm{START} & \rightarrow & \mathrm{MLIST} & \mathrm{ML} & \rightarrow & ml\ \mathrm{MULT}\ \mathrm{MLIST} & & & \\
\mathrm{MLIST} & \rightarrow & mlist & \mathrm{MULT} & \rightarrow & mult & \mathrm{ADD} & \rightarrow & add\\
\mathrm{MLIST} & \rightarrow & mlist\ \mathrm{ML} & \mathrm{MULT} & \rightarrow & mult\ \mathrm{MULT}\ \mathrm{ADD} & \mathrm{ADD} & \rightarrow & add\ \mathrm{ADD}
\end{array}$$

source program

$$
\begin{array}{l}
mlist([\,], I, [\,]).\\
mlist([X|R], I, L) \leftarrow ml(X, R, I, L).\\
ml(X, R, I, [XI|RI]) \leftarrow mult(X, I, XI),\ mlist(R, I, RI).
\end{array}
$$

trace CFG

START → MLIST

MLIST → *mlist*

ML → *ml* MULT MLIST

MULT → *mult*

ADD → *add*

DD

transformed trace CFG

START → MLIST

MLIST → *mlist*

MLIST → *mlist* ML

ML → *ml* MULT MLIST

MULT → *mult*

MULT → *mult* MULT ADD

ADD → *add*

ADD → *add* ADD

- still hard to analyze
- more intuitive?

source program

$$
\begin{array}{l}
\mathit{mlist}([\,], I, [\,]).\\
\mathit{mlist}([X|R], I, L) \leftarrow \mathit{ml}(X, R, I, L).\\
\mathit{ml}(X, R, I, [XI|RI]) \leftarrow \mathit{mult}(X, I, XI),\ \mathit{mlist}(R, I, RI).
\end{array}
$$

trace CFG

$$
\begin{array}{lcl@{\qquad}rcl@{\qquad}rcl}
\mathrm{START} & \longrightarrow & \mathrm{MLIST} & \mathrm{ML} & \longrightarrow & \mathit{ml}\ \mathrm{MULT}\ \mathrm{MLIST} & & & \\
\mathrm{MLIST} & \longrightarrow & \mathit{mlist} & \mathrm{MULT} & \longrightarrow & \mathit{mult} & \mathrm{ADD} & \longrightarrow & \mathit{add}
\end{array}
$$

DD

transformed trace CFG

$$
\begin{array}{lcl@{\qquad}rcl@{\qquad}rcl}
\mathrm{START} & \longrightarrow & \mathrm{MLIST} & \mathrm{ML} & \longrightarrow & \mathit{ml}\ \mathrm{MULT}\ \mathrm{MLIST} & & & \\
\mathrm{MLIST} & \longrightarrow & \mathit{mlist} & \mathrm{MULT} & \longrightarrow & \mathit{mult} & \mathrm{ADD} & \longrightarrow & \mathit{add}\\
\mathrm{MLIST} & \longrightarrow & \mathit{mlist}\ \mathrm{ML} & \mathrm{MULT} & \longrightarrow & \mathit{mult}\ \mathrm{MULT}\ \mathrm{ADD} & \mathrm{ADD} & \longrightarrow & \mathit{add}\ \mathrm{ADD}
\end{array}
$$

- still hard to analyze
- more intuitive?

source program

$$mlist([\,], I, [\,]).$$
$$mlist([X|R], I, L) \leftarrow ml(X, R, I, L).$$
$$ml(X, R, I, [XI|RI]) \leftarrow mult(X, I, XI),\ mlist(R, I, RI).$$

trace CFG

START	→	MLIST	ML	→	*ml* MULT MLIST			
MLIST	→	*mlist*	MULT	→	*mult*	ADD	→	*add*
								DD

transformed trace CFG

START	→	MLIST	ML	→	*ml* MULT MLIST			
MLIST	→	*mlist*	MULT	→	*mult*	ADD	→	*add*
MLIST	→	*mlist* ML	MULT	→	*mult* MULT ADD	ADD	→	*add* ADD

- still hard to analyze
- more intuitive?

Goal: analyze/predict the effects of **partial evaluation** of logic programs

Third try:

- ~~compare the original and the specialized program~~
- ~~approximate the call traces of the original program with a CFG and transform it using the BTA annotations (prediction)~~
- approximate the call traces of the original program with a FA (RE) and transform it using the BTA annotations (prediction)

source program

$$mlist([\,], I, [\,]).$$
$$mlist([X|R], I, L) \leftarrow ml(X, R, I, L).$$
$$ml(X, R, I, [XI|RI]) \leftarrow mult(X, I, XI),\ mlist(R, I, RI).$$
$$mult(0, Y, 0). \quad mult(s(X), Y, Z) \leftarrow mult(X, Y, Z1),\ add(Z1, Y, Z).$$
$$add(X, 0, X). \quad add(X, s(Y), s(Z)) \leftarrow add(X, Y, Z).$$

source program

trace CFG

$$
\begin{array}{rcl@{\qquad}rcl@{\qquad}rcl}
\mathrm{START} & \longrightarrow & \mathrm{MLIST} & \mathrm{ML} & \longrightarrow & \mathit{ml}\ \mathrm{MULT}\ \mathrm{MLIST} & & & \\
\mathrm{MLIST} & \longrightarrow & \mathit{mlist} & \mathrm{MULT} & \longrightarrow & \mathit{mult} & \mathrm{ADD} & \longrightarrow & \mathit{add} \\
\mathrm{MLIST} & \longrightarrow & \mathit{mlist}\ \mathrm{ML} & \mathrm{MULT} & \longrightarrow & \mathit{mult}\ \mathrm{MULT}\ \mathrm{ADD} & \mathrm{ADD} & \longrightarrow & \mathit{add}\ \mathrm{ADD}
\end{array}
$$

source program

trace CFG

trace SRG

$$
\begin{array}{lcl@{\qquad}rcl@{\qquad}lcl}
\mathrm{START} & \longrightarrow & \mathrm{MLIST} & \mathrm{ML} & \longrightarrow & \mathit{ml}\ \mathrm{MULT}\ \mathrm{MLIST} & & & \\
\mathrm{MLIST} & \longrightarrow & \mathit{mlist} & \mathrm{MULT}' & \longrightarrow & \epsilon & \mathrm{ADD} & \longrightarrow & \mathit{add} \\
\mathrm{MLIST} & \longrightarrow & \mathit{mlist}\ \mathrm{ML} & \mathrm{MULT} & \longrightarrow & \mathit{mult}\ \mathrm{MULT}' & \mathrm{ADD} & \longrightarrow & \mathit{add}\ \mathrm{ADD} \\
 & & & \mathrm{MULT} & \longrightarrow & \mathit{mult}\ \mathrm{MULT} & & & \\
 & & & \mathrm{MULT}' & \longrightarrow & \mathrm{ADD}\ \mathrm{MULT}' & & &
\end{array}
$$

[Mohri and Nederhof, Regular Approximation of CFGs through Transformation]

source program

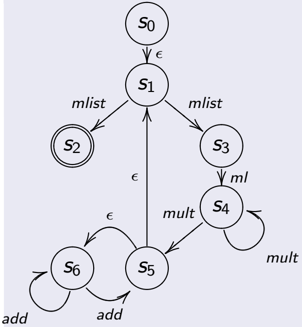
trace CFG

D

trace SRG

$$
\begin{array}{lcllcl}
\mathrm{ML} & \longrightarrow & \mathit{ml}\ \mathrm{MULT}\ \mathrm{MLIST} & & & \\
\mathrm{MULT}' & \longrightarrow & \epsilon & \mathrm{ADD} & \longrightarrow & \mathit{add} \\
\mathrm{MULT} & \longrightarrow & \mathit{mult}\ \mathrm{MULT}' & \mathrm{ADD} & \longrightarrow & \mathit{add}\ \mathrm{ADD} \\
\mathrm{MULT} & \longrightarrow & \mathit{mult}\ \mathrm{MULT} & & & \\
\mathrm{MULT}' & \longrightarrow & \mathrm{ADD}\ \mathrm{MULT}' & & &
\end{array}
$$

trace FA



$s_0 = \mathrm{START},\ \ s_1 = \mathrm{MLIST}, s_2 = \epsilon,$

$s_3 = \mathrm{ML},\ \ s_4 = \mathrm{MULT\ MLIST},$

$s_5 = \mathrm{MULT}'\ \mathrm{MLIST},$

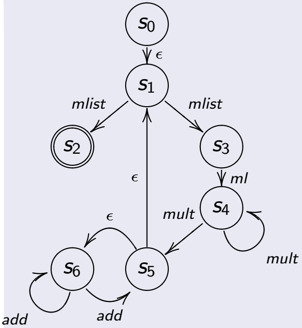
$s_6 = \mathrm{ADD\ MULT}'\ \mathrm{MLIST}$

source program

trace CFG

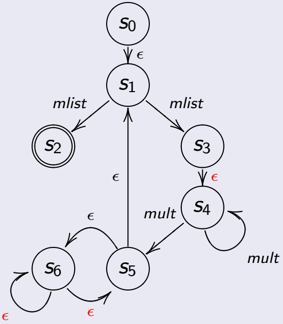
trace SRG

trace FA



$s_0 = \text{START}, \ s_1 = \text{MLIST}, s_2 = \epsilon,$
$s_3 = \text{ML}, \ s_4 = \text{MULT MLIST},$
$s_5 = \text{MULT}' \text{ MLIST},$
$s_6 = \text{ADD MULT}' \text{ MLIST}$

transformed trace FA



$s_0 = \text{START}, \ s_1 = \text{MLIST}, s_2 = \epsilon,$
$s_3 = \text{ML}, \ s_4 = \text{MULT MLIST},$
$s_5 = \text{MULT}' \text{ MLIST},$
$s_6 = \text{ADD MULT}' \text{ MLIST}$

$\text{ADD} \longrightarrow add$
$\text{ADD} \longrightarrow add \ \text{ADD}$

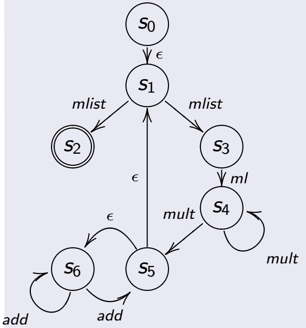
- simpler to analyze !
- more intuitive (hopefully)

source program

trace CFG

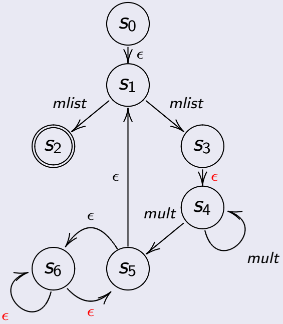
trace SRG

trace FA



s_0 = START, s_1 = MLIST, $s_2 = \epsilon$,
s_3 = ML, s_4 = MULT MLIST,
s_5 = MULT$'$ MLIST,
s_6 = ADD MULT$'$ MLIST

transformed trace FA



s_0 = START, s_1 = MLIST, $s_2 = \epsilon$,
s_3 = ML, s_4 = MULT MLIST,
s_5 = MULT$'$ MLIST,
s_6 = ADD MULT$'$ MLIST

$$\text{ADD} \longrightarrow \mathit{add}$$
$$\text{ADD} \longrightarrow \mathit{add}\ \text{ADD}$$

- simpler to analyze !
- more intuitive (hopefully)

source program

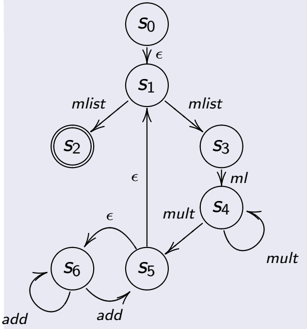
trace CFG

trace SRG

$\text{ADD} \longrightarrow \textit{add}$

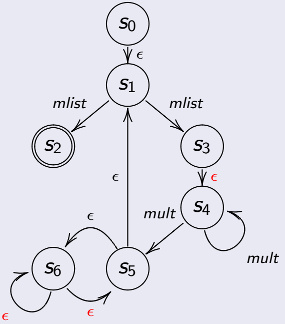
$\text{ADD} \longrightarrow \textit{add}\ \text{ADD}$

trace FA



$s_0 = \text{START},\ \ s_1 = \text{MLIST}, s_2 = \epsilon,$
$s_3 = \text{ML},\ \ s_4 = \text{MULT MLIST},$
$s_5 = \text{MULT}'\ \text{MLIST},$
$s_6 = \text{ADD MULT}'\ \text{MLIST}$

transformed trace FA



$s_0 = \text{START},\ \ s_1 = \text{MLIST}, s_2 = \epsilon,$
$s_3 = \text{ML},\ \ s_4 = \text{MULT MLIST},$
$s_5 = \text{MULT}'\ \text{MLIST},$
$s_6 = \text{ADD MULT}'\ \text{MLIST}$

- simpler to analyze !
- more intuitive (hopefully)

Summary

- new **trace analysis**, useful to represent in a compact (and finite) way the search space
- first **symbolic** approach for predicting the effectivenes of PE
- proof-of-concept implementation PEPE:
  http://german.dsic.upv.es/pepe.html (web interface)

Future work

- more accurate (but still fast) trace analysis (deal with unification, backtracking)
- (semi-)automated (quantitative?) analysis
- . . .

Summary

- new **trace analysis**, useful to represent in a compact (and finite) way the search space
- first **symbolic** approach for predicting the effectivenes of PE
- proof-of-concept implementation PEPE:

  http://german.dsic.upv.es/pepe.html (web interface)

Future work

- more accurate (but still fast) trace analysis (deal with unification, backtracking)
- (semi-)automated (quantitative?) analysis
- . . .