FAKULTÄT FÜR !NFORMATIK
Faculty of !nformatics

Pushing Efficient Evaluation of HEX Programs by Modular Decomposition

Thomas Eiter Michael Fink Giovambattista Ianni
Thomas Krennwallner Peter Schüller

KBS Group – Institut für Informationssysteme, Technische Universität Wien

Dipartimento di Matematica, Università della Calabria

IJCAI – July 20, 2011 – Barcelona

Best Paper Track – LPNMR 2011 – Vancouver

supported by: Austrian Science Fund (FWF) project P20841 and Vienna Science and Technology Fund (WWTF) project ICT08-020

Motivation and Overview

Motivation:

- HEX extends Answer Set Programming
- supports external atoms for external knowledge/computations
- drawbacks of previous HEX evaluation algorithm

$\Rightarrow$ applications called for efficiency improvements

Motivation and Overview

Motivation:

- HEX extends Answer Set Programming
- supports external atoms for external knowledge/computations
- drawbacks of previous HEX evaluation algorithm

⇒ applications called for efficiency improvements

This work:

- novel HEX evaluation formalism

  ⇒ evaluation graph + model graph

- implementation + experimental evaluation

  ⇒ can even outperform standard ASP solvers

Answer Set Programming [Gelfond and Lifschitz, 1991]

Programs are sets of rules of the form

$$\alpha_1 \vee \cdots \vee \alpha_k \leftarrow \beta_1, \ldots, \beta_n, \mathit{not}\ \beta_{n+1}, \ldots, \mathit{not}\ \beta_m$$

where not is negation-as-failure and α_i, β_j are atoms of the form

$$p(t_1, \ldots, t_k).$$

Herbrand base HB_P is the set of all ground atoms using constants of P

Interpretation $I \subseteq HB_P$

$I \models$ rule body iff $\begin{cases} \text{positive atoms } \beta_1, \ldots, \beta_n \text{ are in } I \text{, and} \\ \text{negative atoms } \beta_{n+1}, \ldots, \beta_m \text{ are not in } I \end{cases}$

$I \models$ rule iff some head atom α_i is in I, or the rule body is not satisfied

$I \models P$ iff $I \models grnd(P)$ iff I satisfies all ground rules of program P

FLP reduct [Faber *et al.*, 2011] of P wrt. I:

fP^I is the set of ground rules of P where I satisfies the rule body

I is an answer set of P iff I is a $\subseteq$-minimal model of fP^I

HEX **Syntax and Semantics** **[Eiter *et al.*, 2005]**

Program (choose a plan, choose a usage, determine resource needs):

EDB $\{ \, choose(a, c, d) \leftarrow, \;\; choose(b, e, f) \leftarrow \}$

IDB
$$\left\{ \begin{array}{lrl} r_1\!: plan(a) \vee plan(b) & \leftarrow & \\ r_2\!: & need(p,C) \leftarrow & \&res[plan](C) \\ r_3\!: & use(X) \vee use(Y) \leftarrow & plan(P), choose(P,X,Y) \\ r_4\!: & need(u,C) \leftarrow & \&res[use](C) \\ c_5\!: & \leftarrow & need(\_,money) \end{array} \right\}$$

External Atom:

- $\&g[\mathbf{x}](\mathbf{y})$
- input list $\mathbf{x} = x_1, \ldots, x_n$ and output list $\mathbf{y} = y_1, \ldots, y_m$ of terms
- $I \models \&g[\mathbf{x}](\mathbf{y})$
  iff oracle function $f_{\&g}(I, \mathbf{x}, \mathbf{y})$ is true

HEX **Example**

Program (choose a plan, choose a usage, determine resource needs):

EDB $\{\ choose(a,c,d) \leftarrow,\ \ choose(b,e,f) \leftarrow\ \}$

IDB
$$\left\{\begin{array}{ll} r_1: plan(a) \vee plan(b) \leftarrow & \\ r_2: \quad need(p,C) \leftarrow \&res[plan](C) & \\ r_3: \quad use(X) \vee use(Y) \leftarrow plan(P), choose(P,X,Y) & \\ r_4: \quad need(u,C) \leftarrow \&res[use](C) & \\ c_5: \quad \leftarrow need(\_,money) & \end{array}\right\}$$

External Atom:

- $I \models \&res[p](money)$ if $p(C) \in I$ for $C \in \{a,f\}$,
- $I \models \&res[p](time)$ if $p(C) \in I$ for $C \in \{b,c,d,e\}$,
- $I \not\models \&res[p](X)$ otherwise.

HEX **Example**

Program (choose a plan, choose a usage, determine resource needs):

$$\text{EDB} \quad \{\ choose(a,c,d) \leftarrow, \quad choose(b,e,f) \leftarrow\ \}$$

$$\text{IDB} \quad \left\{ \begin{array}{lrl} r_1: plan(a) \vee plan(b) & \leftarrow & \\ r_2: & need(p,C) \leftarrow & \&res[plan](C) \\ r_3: & use(X) \vee use(Y) \leftarrow & plan(P), choose(P,X,Y) \\ r_4: & need(u,C) \leftarrow & \&res[use](C) \\ c_5: & \leftarrow & need(\_,money) \end{array} \right\}$$

External Atom:

- $I \models \&res[p](money)$ if $p(C) \in I$ for $C \in \{a,f\}$,
- $I \models \&res[p](time)$ if $p(C) \in I$ for $C \in \{b,c,d,e\}$,
- $I \not\models \&res[p](X)$ otherwise.

In our example . . .

$$\{plan(a)\} \models \&res[plan](money) \quad \{plan(a)\} \not\models \&res[plan](time)$$
$$\{plan(b)\} \not\models \&res[plan](money) \quad \{plan(b)\} \models \&res[plan](time)$$

HEX Example

Program (choose a plan, choose a usage, determine resource needs):

EDB $\{\ choose(a,c,d) \leftarrow,\ \ choose(b,e,f) \leftarrow\ \}$

IDB
$$\left\{ \begin{array}{lrl} r_1: plan(a) \vee plan(b) & \leftarrow & \\ r_2: & need(p,C) \leftarrow & \&res[plan](C) \\ r_3: & use(X) \vee use(Y) \leftarrow & plan(P), choose(P,X,Y) \\ r_4: & need(u,C) \leftarrow & \&res[use](C) \\ c_5: & \leftarrow & need(\_,money) \end{array} \right\}$$

Guessed Interpretations:

- $\{plan(a), \&res[plan](money), need(p,money),$
  $use(c), \&res[use](time), need(u,time)\}$
- $\{plan(a), \&res[plan](money), need(p,money),$
  $use(d), \&res[use](time), need(u,time)\}$
- $\{plan(b), \&res[plan](time), need(p,time),$
  $use(e), \&res[use](time), need(u,time)\}$
- $\{plan(b), \&res[plan](time), need(p,time),$
  $use(f), \&res[use](money), need(u,money)\}$

HEX **Example**

Program (choose a plan, choose a usage, determine resource needs):

EDB $\{\ choose(a,c,d) \leftarrow,\ \ choose(b,e,f) \leftarrow\ \}$

IDB
$$\left\{\begin{array}{lrl} r_1\colon plan(a) \vee plan(b) & \leftarrow & \\ r_2\colon & need(p,C) \leftarrow & \&res[plan](C) \\ r_3\colon & use(X) \vee use(Y) \leftarrow & plan(P), choose(P,X,Y) \\ r_4\colon & need(u,C) \leftarrow & \&res[use](C) \\ c_5\colon & \leftarrow & need(\_,money) \end{array}\right\}$$

Guessed Interpretations:

- ~~$\{plan(a), \&res[plan](money), need(p, money),$ $use(c), \&res[use](time), need(u, time)\}$~~
- ~~$\{plan(a), \&res[plan](money), need(p, money),$ $use(d), \&res[use](time), need(u, time)\}$~~
- $\{plan(b), \&res[plan](time), need(p, time),$ $use(e), \&res[use](time), need(u, time)\}$ $\Leftarrow$ Answer Set
- ~~$\{plan(b), \&res[plan](time), need(p, time),$ $use(f), \&res[use](money), need(u, money)\}$~~

HEX evaluation (previous strategy)

(1) calculate models of all program parts that do not depend on external computations that still need to be done

(2) do external computations that depend on (1)

(3) replace evaluated external atoms by result of (2) and restart at (1)

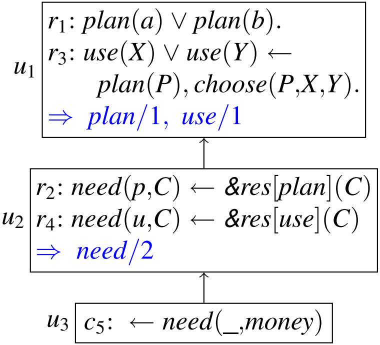
HEX **evaluation (previous strategy)**

(1) calculate models of all program parts that do not depend on external computations that still need to be done

(2) do external computations that depend on (1)

(3) replace evaluated external atoms by result of (2) and restart at (1)

$$
\begin{aligned}
&r_1: plan(a) \vee plan(b).\\
&r_2: need(p,C) \leftarrow\\
&\qquad \&res[plan](C).\\
&r_3: use(X) \vee use(Y) \leftarrow\\
&\qquad plan(P), choose(P,X,Y).\\
&r_4: need(u,C) \leftarrow\\
&\qquad \&res[use](C).\\
&c_5: \leftarrow need(\_,money).
\end{aligned}
$$



HEX **model building (previous strategy)**

"Evaluation Graph"

r_1: $plan(a) \vee plan(b).$
r_3: $use(X) \vee use(Y) \leftarrow$
$plan(P), choose(P,X,Y).$

r_2: $need(p,C) \leftarrow$
$\&res[plan](C)$
r_4: $need(u,C) \leftarrow$
$\&res[use](C)$

c_5: $\leftarrow need(\_,money)$

Models

$m_1 \stackrel{o:-}{=} \{plan(a), use(c)\}$
$m_2 \stackrel{o:-}{=} \{plan(a), use(d)\}$
$m_3 \stackrel{o:-}{=} \{plan(b), use(e)\}$
$m_4 \stackrel{o:-}{=} \{plan(b), use(f)\}$

$m_5 \stackrel{i:1}{=} m_1$ $m_6 \stackrel{i:2}{=} m_2$
$m_7 \stackrel{i:3}{=} m_3$ $m_8 \stackrel{i:4}{=} m_4$

HEX **model building (previous strategy)**

“Evaluation Graph”

$r_1\colon plan(a) \lor plan(b).$
$r_3\colon use(X) \lor use(Y) \leftarrow plan(P), choose(P,X,Y).$

↑

$r_2\colon need(p,C) \leftarrow \&res[plan](C)$
$r_4\colon need(u,C) \leftarrow \&res[use](C)$

↑

$c_5\colon \leftarrow need(\_,money)$

Models

$m_1 \overset{\text{O:}-}{=} \{plan(a), use(c)\}$
$m_2 \overset{\text{O:}-}{=} \{plan(a), use(d)\}$
$m_3 \overset{\text{O:}-}{=} \{plan(b), use(e)\}$
$m_4 \overset{\text{O:}-}{=} \{plan(b), use(f)\}$

↑

$m_5 \overset{\text{I:}1}{=} m_1$ $\quad m_6 \overset{\text{I:}2}{=} m_2$
$m_7 \overset{\text{I:}3}{=} m_3$ $\quad m_8 \overset{\text{I:}4}{=} m_4$
$m_9 \overset{\text{O:}5}{=} \{need(p,money), need(u,time)\}$
$m_{10} \overset{\text{O:}6}{=} \{need(p,money), need(u,time)\}$
$m_{11} \overset{\text{O:}7}{=} \{need(p,time), need(u,time)\}$
$m_{12} \overset{\text{O:}8}{=} \{need(p,time), need(u,money)\}$

↑

$m_{13} \overset{\text{I:}9}{=} m_9$ $\quad m_{14} \overset{\text{I:}10}{=} m_{10}$
$m_{15} \overset{\text{I:}11}{=} m_{11}$ $\quad m_{16} \overset{\text{I:}12}{=} m_{12}$

HEX **model building (previous strategy)**

"Evaluation Graph"

r_1: $plan(a) \lor plan(b)$.
r_3: $use(X) \lor use(Y) \leftarrow$
$plan(P), choose(P,X,Y)$.

↑

r_2: $need(p,C) \leftarrow$
$\&res[plan](C)$
r_4: $need(u,C) \leftarrow$
$\&res[use](C)$

↑

c_5: $\leftarrow need(\_,money)$

Models

$m_1 \overset{\text{O:-}}{=} \{plan(a), use(c)\}$
$m_2 \overset{\text{O:-}}{=} \{plan(a), use(d)\}$
$m_3 \overset{\text{O:-}}{=} \{plan(b), use(e)\}$
$m_4 \overset{\text{O:-}}{=} \{plan(b), use(f)\}$

↑

$m_5 \overset{\text{I:1}}{=} m_1$ $\quad m_6 \overset{\text{I:2}}{=} m_2$
$m_7 \overset{\text{I:3}}{=} m_3$ $\quad m_8 \overset{\text{I:4}}{=} m_4$
$m_9 \overset{\text{O:5}}{=} \{need(p,money), need(u,time)\}$
$m_{10} \overset{\text{O:6}}{=} \{need(p,money), need(u,time)\}$
$m_{11} \overset{\text{O:7}}{=} \{need(p,time), need(u,time)\}$
$m_{12} \overset{\text{O:8}}{=} \{need(p,time), need(u,money)\}$

↑

$m_{13} \overset{\text{I:9}}{=} m_9$ $\quad m_{14} \overset{\text{I:10}}{=} m_{10}$
$m_{15} \overset{\text{I:11}}{=} m_{11}$ $\quad m_{16} \overset{\text{I:12}}{=} m_{12}$
$m_{17} \overset{\text{O:15}}{=} \emptyset$

HEX **model building (previous strategy)**

"Evaluation Graph"

r_1: $plan(a) \vee plan(b).$
r_3: $use(X) \vee use(Y) \leftarrow$
$plan(P), choose(P,X,Y).$

$\uparrow$

r_2: $need(p,C) \leftarrow$
$\&res[plan](C)$
r_4: $need(u,C) \leftarrow$
$\&res[use](C)$

$\uparrow$

c_5: $\leftarrow need(\_,money)$

Models

$m_1 \overset{\text{o:}-}{=} \{plan(a), use(c)\}$
$m_2 \overset{\text{o:}-}{=} \{plan(a), use(d)\}$
$m_3 \overset{\text{o:}-}{=} \{plan(b), use(e)\}$
$m_4 \overset{\text{o:}-}{=} \{plan(b), use(f)\}$

$\uparrow$

$m_5 \overset{\text{i:1}}{=} m_1$ $m_6 \overset{\text{i:2}}{=} m_2$
$m_7 \overset{\text{i:3}}{=} m_3$ $m_8 \overset{\text{i:4}}{=} m_4$
$m_9 \overset{\text{o:5}}{=} \{need(p,money), need(u,time)\}$
$m_{10} \overset{\text{o:6}}{=} \{need(p,money), need(u,time)\}$
$m_{11} \overset{\text{o:7}}{=} \{need(p,time), need(u,time)\}$
$m_{12} \overset{\text{o:8}}{=} \{need(p,time), need(u,money)\}$

$\uparrow$

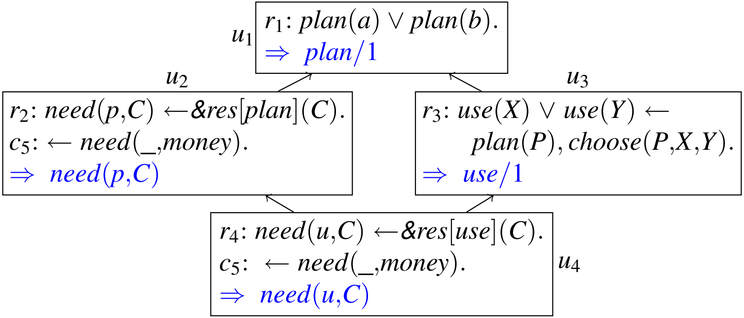
$m_{13} \overset{\text{i:9}}{=} m_9$ $m_{14} \overset{\text{i:10}}{=} m_{10}$
$m_{15} \overset{\text{i:11}}{=} m_{11}$ $m_{16} \overset{\text{i:12}}{=} m_{12}$
$m_{17} \overset{\text{o:15}}{=} \emptyset$

Observations

- We compute the largest possible fragment
  ⇒ multiplication of independent guesses
  ⇒ redundant evaluation of external atoms
- Constraints are considered if all dependencies are fulfilled
  ⇒ some constraints could safely eliminate models earlier
- We calculate all models at one unit, then go to the next unit
  ⇒ if we need only one model, we calculate unnecessary models

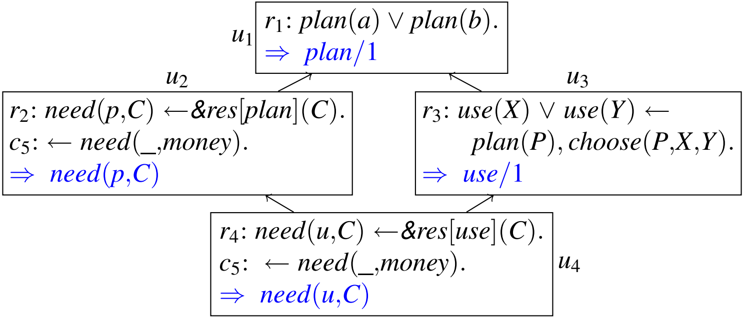
⇒ **Several improvements possible!**

Improved Strategy Evaluation Graph



- Acyclic Evaluation Graph
- Rules cannot be shared among units
- Constraints can be shared among units
- Rule dependencies must be fulfilled
- Constraint dependencies (less strict) must be fulfilled
  $\Rightarrow$ e.g., c_5 has no negative body, therefore it can be freely shared

Improved Strategy **Model Building**

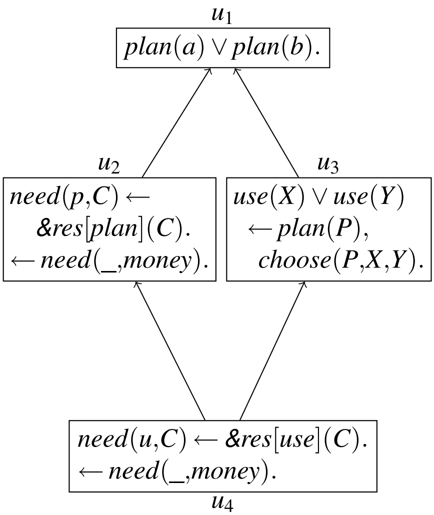


Input and Output models at each Evaluation Unit:

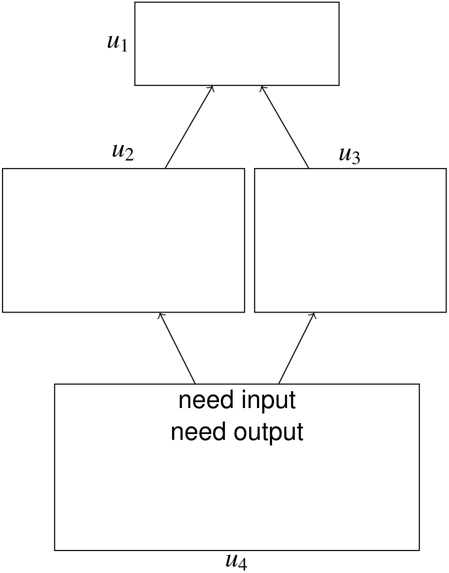
- output model $=$ evaluation of program fragment on input model
- input model $=$ joined output model of predecessor unit(s)
- Principle: common ancestor model at common ancestor units

> Soundness + Completeness Theorem:
> I = union of connected output models at every unit
> iff I = answer set of the program

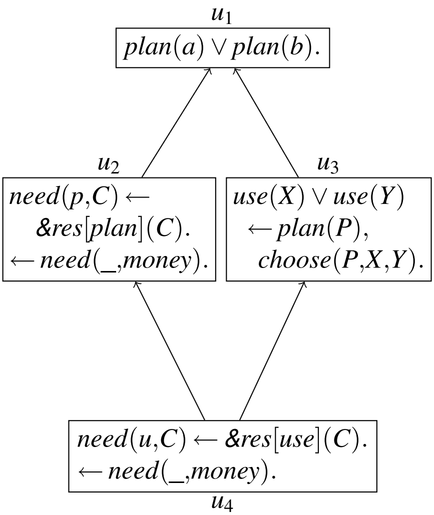
Improved Strategy Model Building



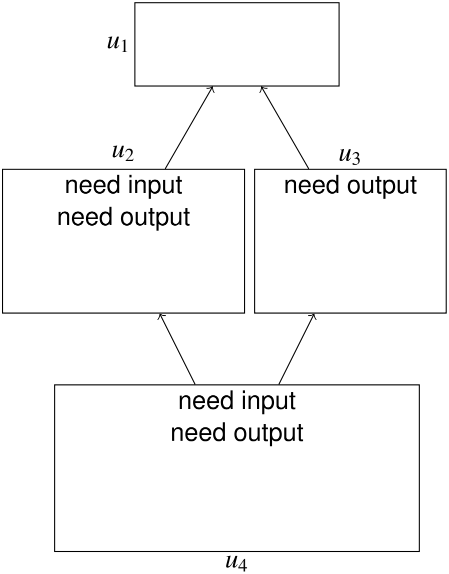
Model Graph



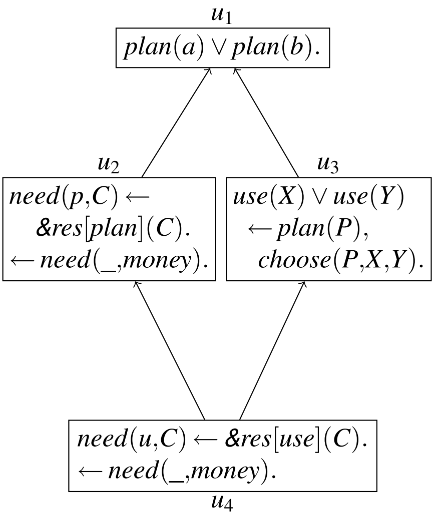
Improved Strategy Model Building



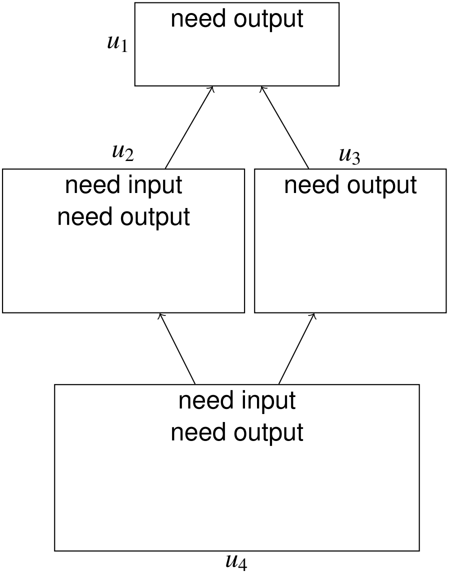
Model Graph



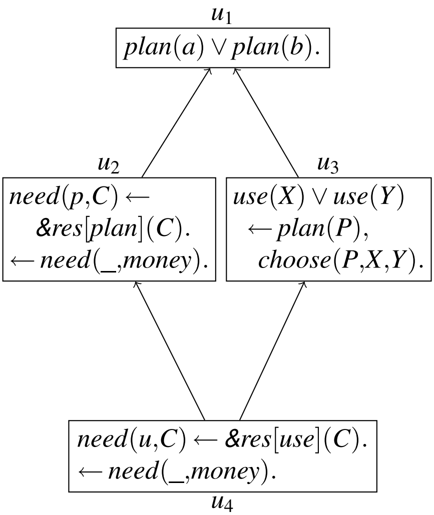
Improved Strategy Model Building



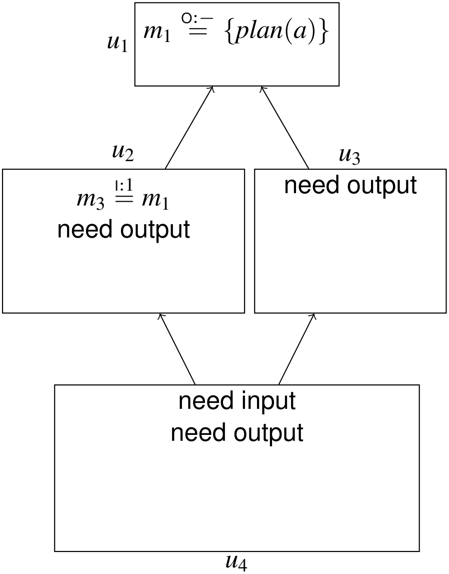
Model Graph



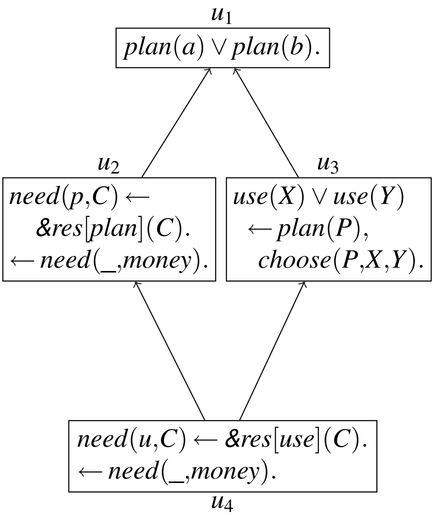
Improved Strategy Model Building



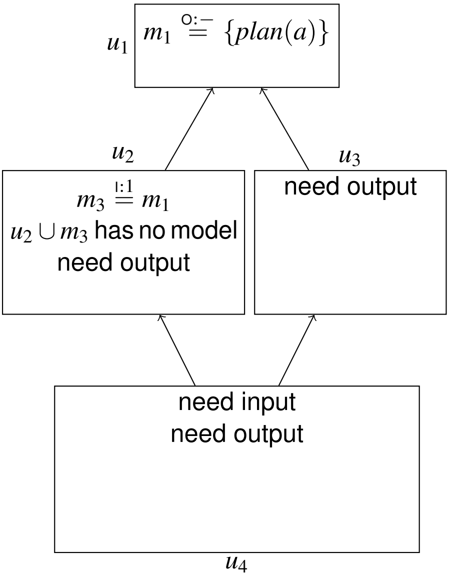
Model Graph



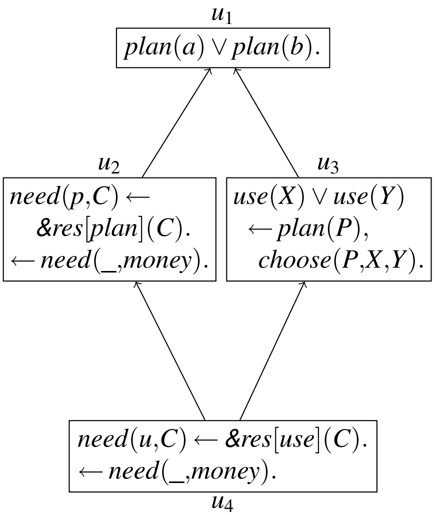
Improved Strategy Model Building



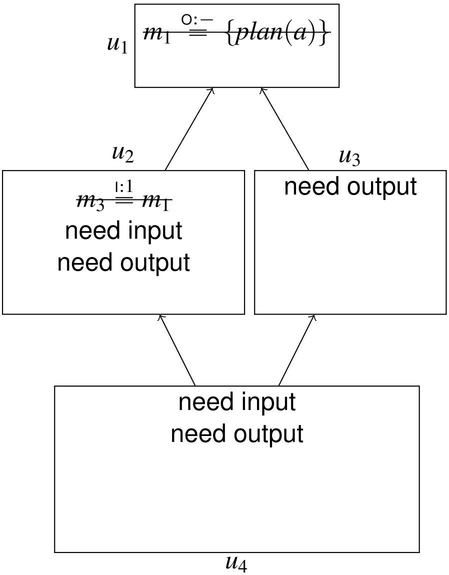
Model Graph



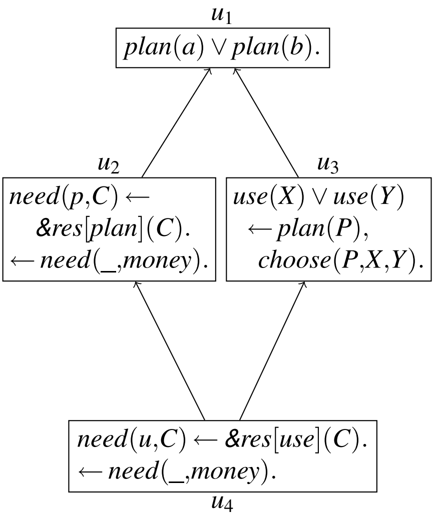
Improved Strategy Model Building



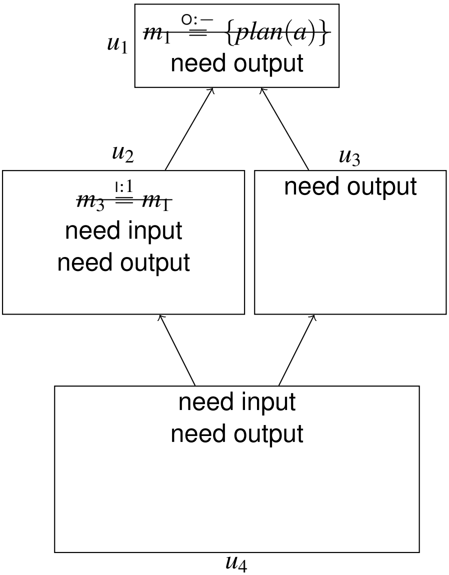
Model Graph



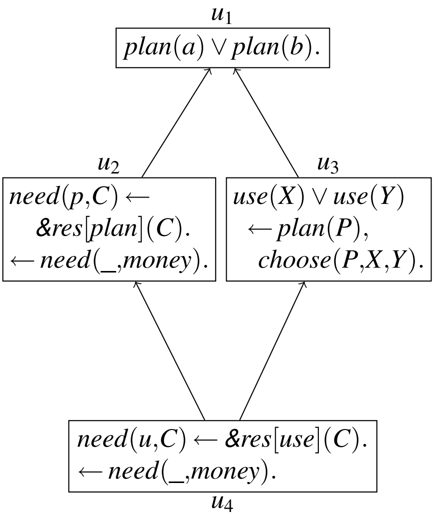
Improved Strategy Model Building



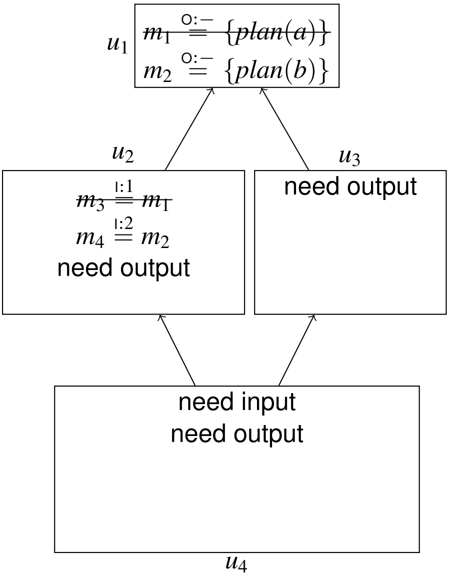
Model Graph



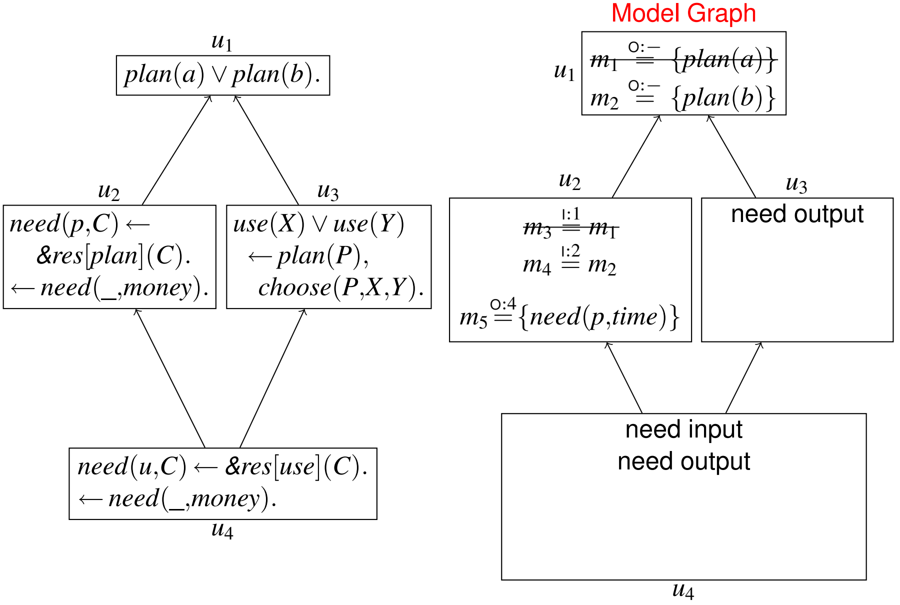
Improved Strategy Model Building



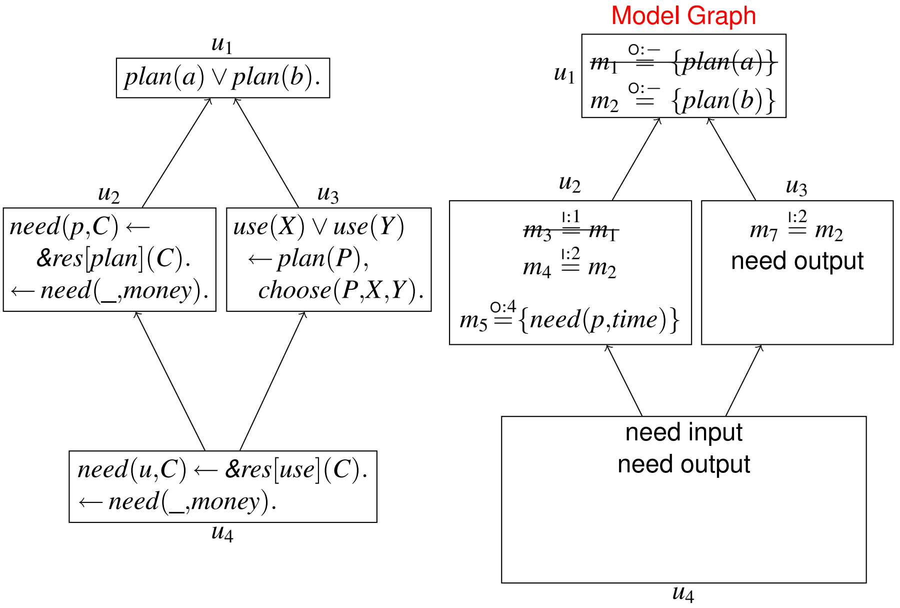
Model Graph



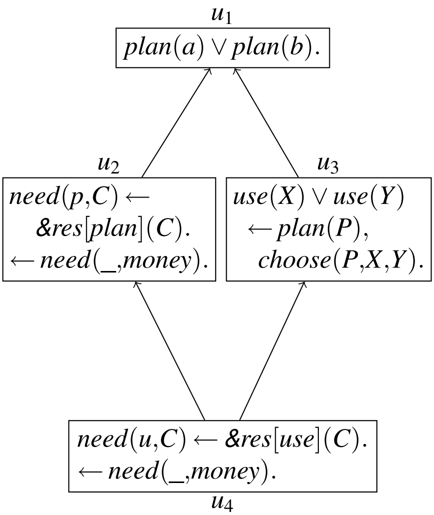
Improved Strategy Model Building



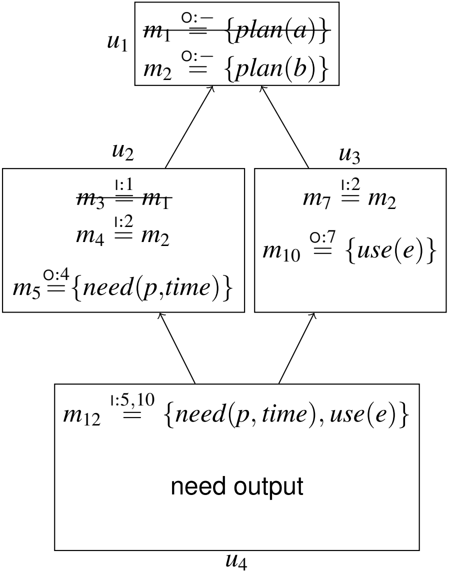
Improved Strategy Model Building



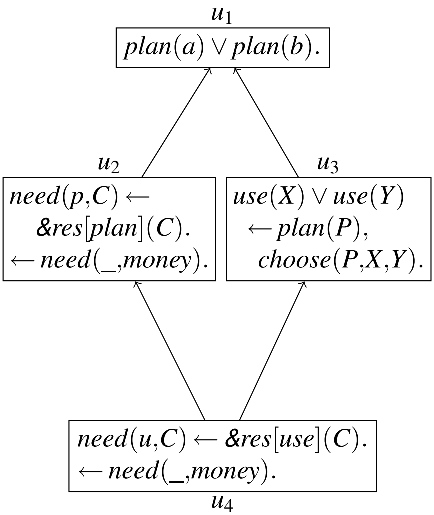
Improved Strategy Model Building

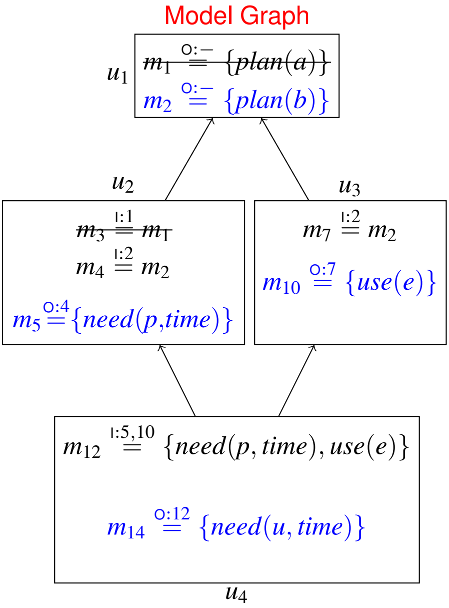


Model Graph

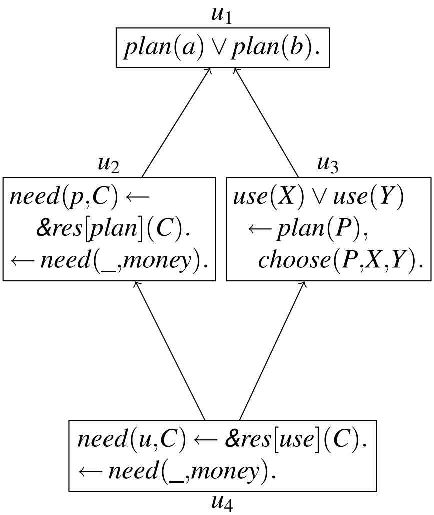


Improved Strategy Model Building

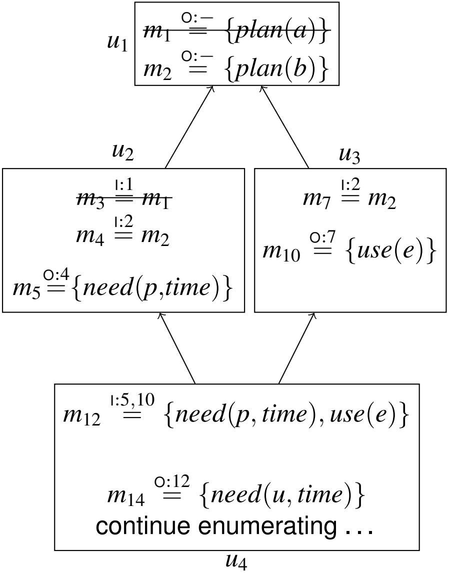




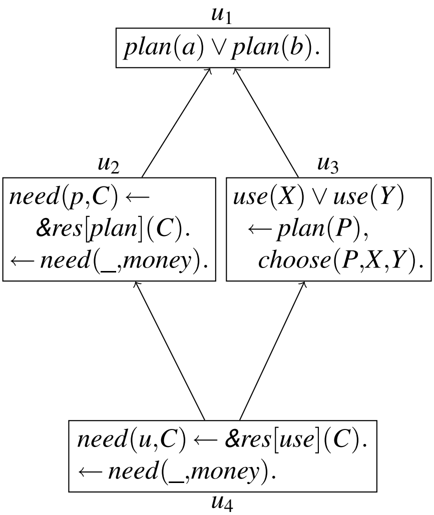
Improved Strategy Model Building



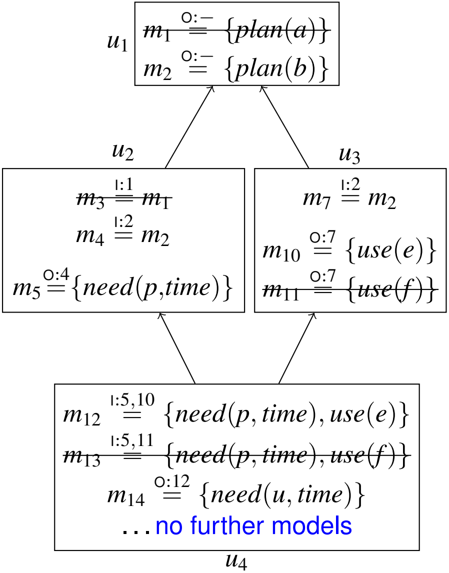
Model Graph



Improved Strategy Model Building



Model Graph



Improvements

Acyclic Directed Evaluation Graph

- ⇒ independent program fragments evaluated independently
- ⇒ fewer redundant evaluations
- ⇒ parallel evaluation possible

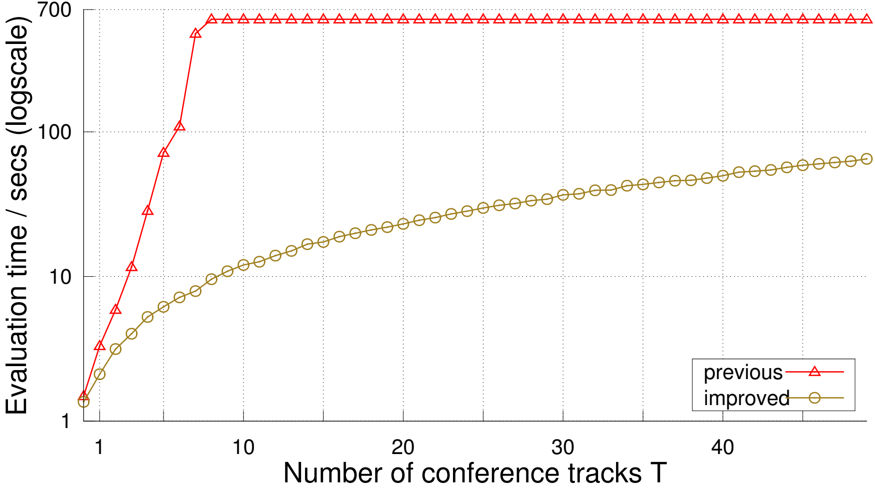
Constraints can be shared between units

- ⇒ early model elimination

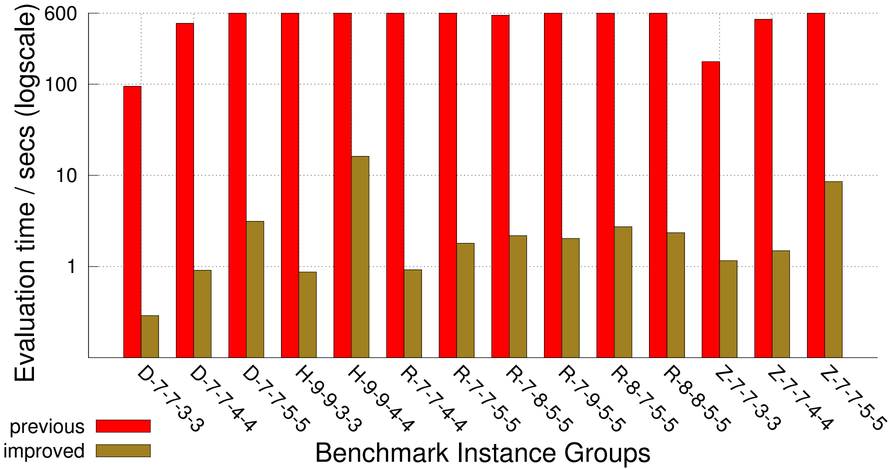
Calculating one model at a time

- ⇒ first model obtained faster
- ⇒ can discard some intermediate results early

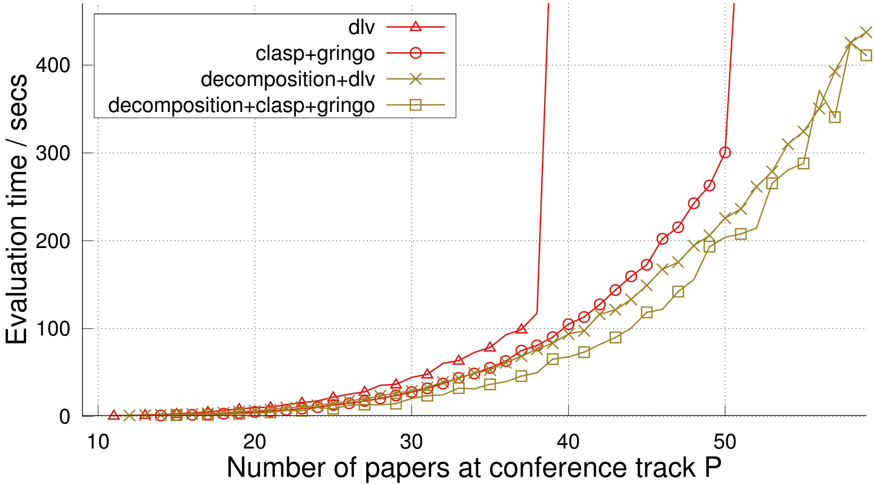
Reviewer Selection



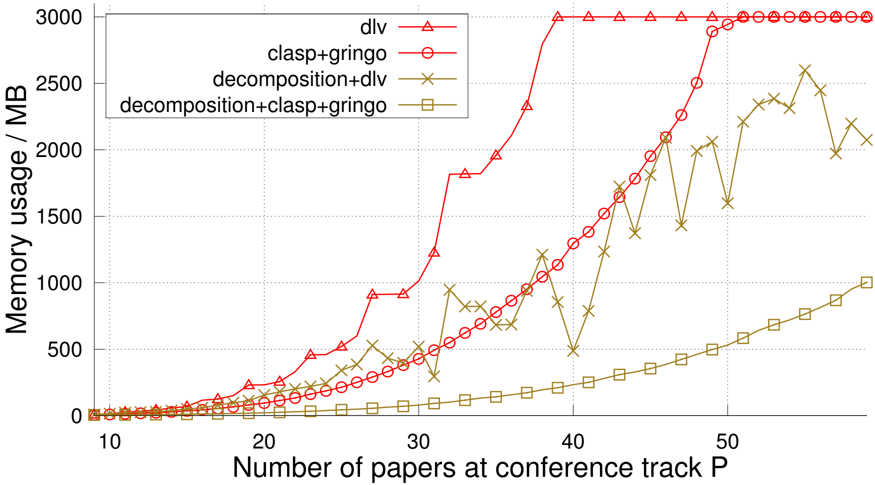
Multi-Context Systems Evaluation



Reviewer Selection Plain ASP



Reviewer Selection Plain ASP



Conclusion

Novel Theoretical HEX Evaluation Framework

⇒ Evaluation Graph + Model Graph

⇒ exponential speedup for certain instances

⇒ potential for optimization strategies
(emphasis on parallelization vs. memory usage)

⇒ can even outperform native ASP solvers in some cases

Improved Implementation (Open Source)

⇒ backends `dlv` and `clingo`

⇒ Instructions for building and detailed experimental results:
click `Experiments` on the `dlvhex` homepage
`http://www.kr.tuwien.ac.at/research/systems/dlvhex/`

References I

- Thomas Eiter, Giovambattista Ianni, Roman Schindlauer, and Hans Tompits. A Uniform Integration of Higher-Order Reasoning and External Evaluations in Answer-Set Programming. In *IJCAI-05*, pages 90–96, 2005.
- Wolfgang Faber, Nicola Leone, and Gerald Pfeifer. Semantics and complexity of recursive aggregates in answer set programming. *Artificial Intelligence*, 175(1):278–298, January 2011.
- Michael Gelfond and Vladimir Lifschitz. Classical Negation in Logic Programs and Disjunctive Databases. *Next Generation Computing*, 9(3–4):365–386, 1991.