

# Pushing Efficient Evaluation of HEX Programs by Modular Decomposition

Thomas Eiter   Michael Fink  
Thomas Krennwallner

Giovambattista Ianni  
Peter Schüller



KBS Group – Institut für Informationssysteme, Technische Universität Wien  
Dipartimento di Matematica, Università della Calabria

LPNMR – May 18, 2011

- ▶ Answer Set Programming ‘on one slide’
- ▶ HEX basics
- ▶ Old HEX evaluation Example
- ▶ Improved HEX evaluation Example
- ▶ Formalism Improvements
- ▶ Implementation Improvements
- ▶ Benchmark Results

Atoms of the form  $p(t_1, \dots, t_k)$ .

Rules of the form  $\alpha_1 \vee \dots \vee \alpha_k \leftarrow \beta_1, \dots, \beta_n, \text{not } \beta_{n+1}, \dots, \text{not } \beta_m$ .

Program  $P$  is a set of rules.

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

Interpretation  $I \subseteq HB_P$ .

$I \models \text{grnd}(P)$  iff  $I$  satisfies all ground rules.

$I \models$  rule body iff positive atoms are in  $I$  and negative atoms are not.

$I \models$  rule iff some  $\alpha_i \in I$ , or  $I$  does not satisfy the rule body.

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 iff  $I$  is a minimal model of  $fP^I$ .

Example Program (IDB):

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

External Atom:

- ▶  $\&g[\mathbf{x}](\mathbf{y})$
- ▶ with input list  $\mathbf{x} = x_1, \dots, x_n$  and output list  $\mathbf{y} = y_1, \dots, y_m$

Example Program (IDB):

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

External Atom:

- ▶  $\&g[\mathbf{x}](\mathbf{y})$
- ▶ with input list  $\mathbf{x} = x_1, \dots, x_n$  and output list  $\mathbf{y} = y_1, \dots, y_m$
- ▶  $I \models \&g[\mathbf{x}](\mathbf{y})$   
iff **oracle function**  $f_{\&g}(I, \mathbf{x}, \mathbf{y})$  is true
- ▶ intuitively:  $f_{\&g}$  is true iff  $\mathbf{y}$  is an output of  $\&g$  for input  $I$  and  $\mathbf{x}$

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

Example Program (IDB):

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

External Atom:

- ▶  $\&cost[p](money)$  is true if  $p/1$  is true for  $a$  or  $f$ ,
- ▶  $\&cost[p](time)$  is true if  $p/1$  is true for  $b, c, d$ , or  $e$ ,
- ▶  $\&cost[p](X)$  is false for all other  $X$ -es.

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

Example Program (IDB):

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

External Atom:

- ▶  $\&cost[p](money)$  is true if  $p/1$  is true for  $a$  or  $f$ ,
- ▶  $\&cost[p](time)$  is true if  $p/1$  is true for  $b, c, d$ , or  $e$ ,
- ▶  $\&cost[p](X)$  is false for all other  $X$ -es.

In our example ...

$$\begin{array}{l} \{plan(a)\} \models \&cost[plan](money) \quad \{plan(a)\} \not\models \&cost[plan](time) \\ \{plan(b)\} \not\models \&cost[plan](money) \quad \{plan(b)\} \models \&cost[plan](time) \end{array}$$

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

Example Program (IDB):

$$\left\{ \begin{array}{l} r_1: plan(a) \vee plan(b) \leftarrow \\ r_2: \quad \quad \quad need(p, C) \leftarrow \&cost[plan](C) \\ r_3: use(X) \vee use(Y) \leftarrow plan(P), choose(P, X, Y) \\ r_4: \quad \quad \quad need(u, C) \leftarrow \&cost[use](C) \\ r_5: \quad \quad \quad \quad \quad \quad \leftarrow need(\_, money) \end{array} \right\}$$

Answer Set Candidates (Guesses, without EDB):

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

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

Example Program (IDB):

$$\left\{ \begin{array}{l} r_1: plan(a) \vee plan(b) \leftarrow \\ r_2: \quad \quad \quad need(p, C) \leftarrow \&cost[plan](C) \\ r_3: use(X) \vee use(Y) \leftarrow plan(P), choose(P, X, Y) \\ r_4: \quad \quad \quad need(u, C) \leftarrow \&cost[use](C) \\ r_5: \quad \quad \quad \quad \quad \quad \leftarrow need(\_, money) \end{array} \right\}$$

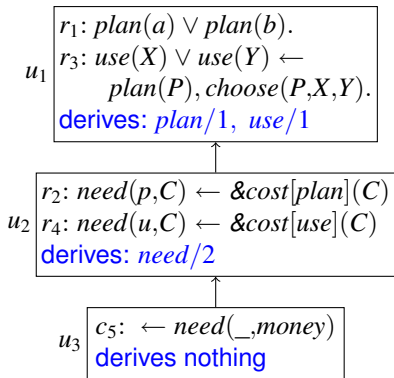
Answer Set Candidates (Guesses, without EDB):

- ▶  $\{plan(a), \&cost[plan](money), \color{red}need(p, money),$   
 $\color{gray}\text{--}use(c), \&cost[use](time), need(u, time)\}$
- ▶  $\{plan(a), \&cost[plan](money), \color{red}need(p, money),$   
 $\color{gray}\text{--}use(d), \&cost[use](time), need(u, time)\}$
- ▶  $\{plan(b), \&cost[plan](time), need(p, time), \quad \leftarrow \text{Answer Set}$   
 $use(e), \&cost[use](time), need(u, time)\}$
- ▶  $\{plan(b), \&cost[plan](time), need(p, time),$   
 $\color{gray}\text{--}use(f), \&cost[use](money), \color{red}need(u, money)\}$

- (1) calculate models of largest possible program part which does not depend on (not yet calculated) external atoms
- (2) calculate external atoms that depend on (1)
- (3) replace external atoms by result of (2) and goto (1)

- (1) calculate models of largest possible program part which does not depend on (not yet calculated) external atoms
- (2) calculate external atoms that depend on (1)
- (3) replace external atoms by result of (2) and goto (1)

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



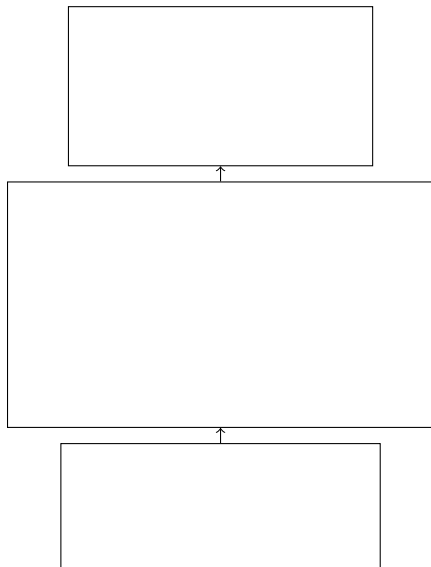
## “Evaluation Graph”

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

$r_2: \text{need}(p, C) \leftarrow$   
 $\quad \&\text{cost}[\text{plan}](C)$   
 $r_4: \text{need}(u, C) \leftarrow$   
 $\quad \&\text{cost}[\text{use}](C)$

$c_5: \leftarrow \text{need}(\_, \text{money})$

## “Model Graph”



“Evaluation Graph”

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

$r_2: \text{need}(p, C) \leftarrow$   
 $\quad \&\text{cost}[\text{plan}](C)$   
 $r_4: \text{need}(u, C) \leftarrow$   
 $\quad \&\text{cost}[\text{use}](C)$

$c_5: \leftarrow \text{need}(\_, \text{money})$

“Model Graph”

$m_1 \stackrel{0:-}{=} \{\text{plan}(a), \text{use}(c)\}$

$m_2 \stackrel{0:-}{=} \{\text{plan}(a), \text{use}(d)\}$

$m_3 \stackrel{0:-}{=} \{\text{plan}(b), \text{use}(e)\}$

$m_4 \stackrel{0:-}{=} \{\text{plan}(b), \text{use}(f)\}$

$m_5 \stackrel{!:1}{=} m_1 \quad m_6 \stackrel{!:2}{=} m_2$

$m_7 \stackrel{!:3}{=} m_3 \quad m_8 \stackrel{!:4}{=} m_4$

“Evaluation Graph”

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

$r_2: \text{need}(p, C) \leftarrow$   
 $\quad \&\text{cost}[\text{plan}](C)$   
 $r_4: \text{need}(u, C) \leftarrow$   
 $\quad \&\text{cost}[\text{use}](C)$

$c_5: \leftarrow \text{need}(\_, \text{money})$

“Model Graph”

$m_1 \stackrel{0:-}{=} \{ \text{plan}(a), \text{use}(c) \}$

$m_2 \stackrel{0:-}{=} \{ \text{plan}(a), \text{use}(d) \}$

$m_3 \stackrel{0:-}{=} \{ \text{plan}(b), \text{use}(e) \}$

$m_4 \stackrel{0:-}{=} \{ \text{plan}(b), \text{use}(f) \}$

$m_5 \stackrel{1:1}{=} m_1 \quad m_6 \stackrel{1:2}{=} m_2$

$m_7 \stackrel{1:3}{=} m_3 \quad m_8 \stackrel{1:4}{=} m_4$

$m_9 \stackrel{0:5}{=} \{ \text{need}(p, \text{money}), \text{need}(u, \text{time}) \}$

$m_{10} \stackrel{0:6}{=} \{ \text{need}(p, \text{money}), \text{need}(u, \text{time}) \}$

$m_{11} \stackrel{0:7}{=} \{ \text{need}(p, \text{time}), \text{need}(u, \text{time}) \}$

$m_{12} \stackrel{0:8}{=} \{ \text{need}(p, \text{time}), \text{need}(u, \text{money}) \}$

$m_{13} \stackrel{1:9}{=} m_9 \quad m_{14} \stackrel{1:10}{=} m_{10}$

$m_{15} \stackrel{1:11}{=} m_{11} \quad m_{16} \stackrel{1:12}{=} m_{12}$

“Evaluation Graph”

$$r_1: \text{plan}(a) \vee \text{plan}(b).$$

$$r_3: \text{use}(X) \vee \text{use}(Y) \leftarrow$$

$$\text{plan}(P), \text{choose}(P, X, Y).$$

$$r_2: \text{need}(p, C) \leftarrow$$

$$\&\text{cost}[\text{plan}](C)$$

$$r_4: \text{need}(u, C) \leftarrow$$

$$\&\text{cost}[\text{use}](C)$$

$$c_5: \leftarrow \text{need}(\_, \text{money})$$

“Model Graph”

$$m_1 \stackrel{0:-}{=} \{\text{plan}(a), \text{use}(c)\}$$

$$m_2 \stackrel{0:-}{=} \{\text{plan}(a), \text{use}(d)\}$$

$$m_3 \stackrel{0:-}{=} \{\text{plan}(b), \text{use}(e)\}$$

$$m_4 \stackrel{0:-}{=} \{\text{plan}(b), \text{use}(f)\}$$

$$m_5 \stackrel{1:1}{=} m_1 \quad m_6 \stackrel{1:2}{=} m_2$$

$$m_7 \stackrel{1:3}{=} m_3 \quad m_8 \stackrel{1:4}{=} m_4$$

$$m_9 \stackrel{0:5}{=} \{\text{need}(p, \text{money}), \text{need}(u, \text{time})\}$$

$$m_{10} \stackrel{0:6}{=} \{\text{need}(p, \text{money}), \text{need}(u, \text{time})\}$$

$$m_{11} \stackrel{0:7}{=} \{\text{need}(p, \text{time}), \text{need}(u, \text{time})\}$$

$$m_{12} \stackrel{0:8}{=} \{\text{need}(p, \text{time}), \text{need}(u, \text{money})\}$$

$$m_{13} \stackrel{1:9}{=} m_9 \quad m_{14} \stackrel{1:10}{=} m_{10}$$

$$m_{15} \stackrel{1:11}{=} m_{11} \quad m_{16} \stackrel{1:12}{=} m_{12}$$

$$m_{17} \stackrel{0:15}{=} \emptyset$$

“Evaluation Graph”

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

$r_2: \text{need}(p, C) \leftarrow$   
 $\text{\&cost}[\text{plan}](C)$   
 $r_4: \text{need}(u, C) \leftarrow$   
 $\text{\&cost}[\text{use}](C)$

$c_5: \leftarrow \text{need}(\_, \text{money})$

“Model Graph”

$m_1 \stackrel{0:-}{=} \{ \text{plan}(a), \text{use}(c) \}$

$m_2 \stackrel{0:-}{=} \{ \text{plan}(a), \text{use}(d) \}$

$m_3 \stackrel{0:-}{=} \{ \text{plan}(b), \text{use}(e) \}$

$m_4 \stackrel{0:-}{=} \{ \text{plan}(b), \text{use}(f) \}$

$m_5 \stackrel{1:1}{=} m_1$        $m_6 \stackrel{1:2}{=} m_2$

$m_7 \stackrel{1:3}{=} m_3$        $m_8 \stackrel{1:4}{=} m_4$

$m_9 \stackrel{0:5}{=} \{ \text{need}(p, \text{money}), \text{need}(u, \text{time}) \}$

$m_{10} \stackrel{0:6}{=} \{ \text{need}(p, \text{money}), \text{need}(u, \text{time}) \}$

$m_{11} \stackrel{0:7}{=} \{ \text{need}(p, \text{time}), \text{need}(u, \text{time}) \}$

$m_{12} \stackrel{0:8}{=} \{ \text{need}(p, \text{time}), \text{need}(u, \text{money}) \}$

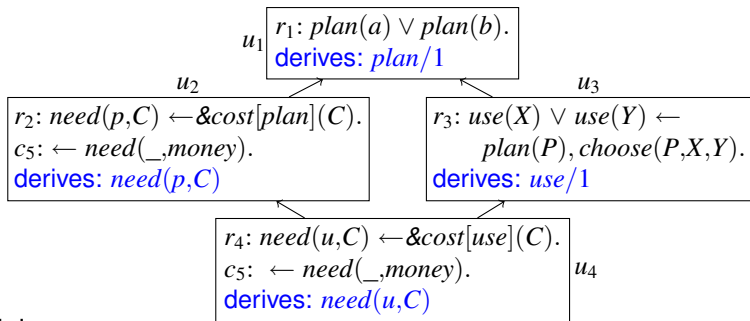
$m_{13} \stackrel{1:9}{=} m_9$        $m_{14} \stackrel{1:10}{=} m_{10}$

$m_{15} \stackrel{1:11}{=} m_{11}$        $m_{16} \stackrel{1:12}{=} m_{12}$

$m_{17} \stackrel{0:15}{=} \emptyset$

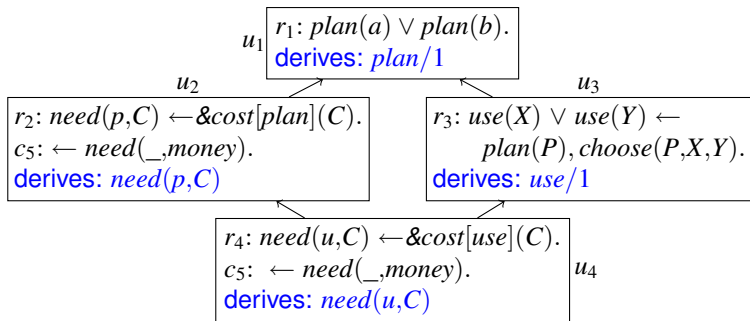


- ▶ We always compute the **largest possible fragment**.
    - ⇒ multiplication of independent guesses
    - ⇒ redundant evaluation of external atoms
  - ▶ Constraints are considered once **all dependencies** are fulfilled
    - ⇒ some models are eliminated later than possible
  - ▶ We calculate **all models** at one unit, then go to the next unit.
    - ⇒ if we need only one model, we calculate unnecessary models
- ⇒ **Many improvements possible!**



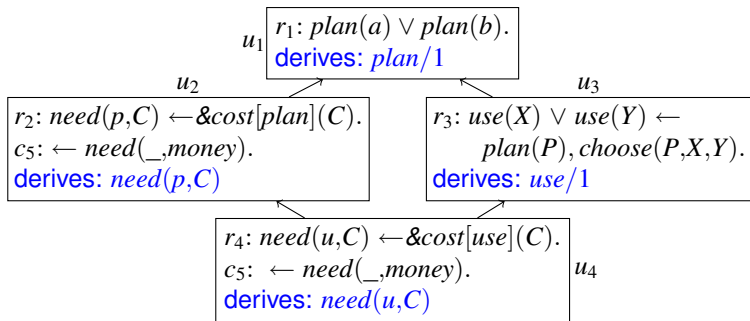
## Principles:

- ▶ **Acyclic** Evaluation Graph
- ▶ Evaluation Units derive **disjoint** atoms
  - $\Rightarrow$  constraints can be in multiple units, rules cannot be
- ▶ Rule dependencies fulfilled everywhere
- ▶ Negative constraint dependencies fulfilled everywhere
- ▶ Positive constraint dependencies fulfilled at least once



Input and Output models at Evaluation Units:

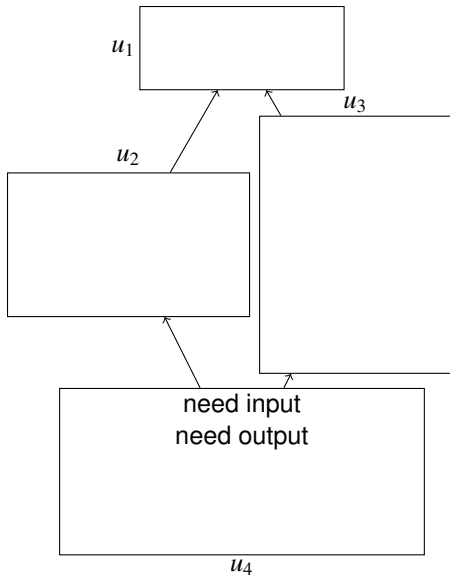
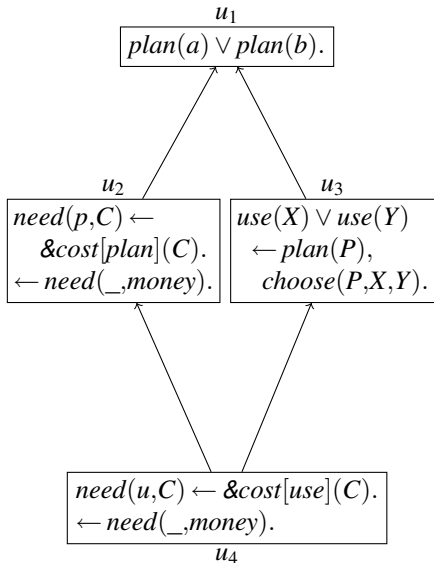
- ▶ input models  $\Leftarrow$  joining output models of predecessor units
- ▶ output models  $\Leftarrow$  evaluating program fragment on input models
- ▶ Principle: **same ancestor** at common ancestor units

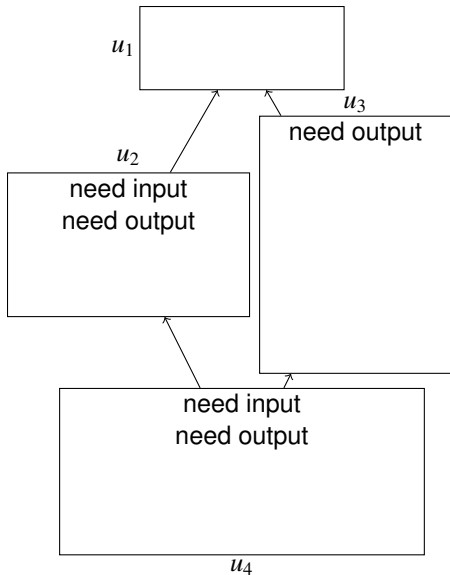
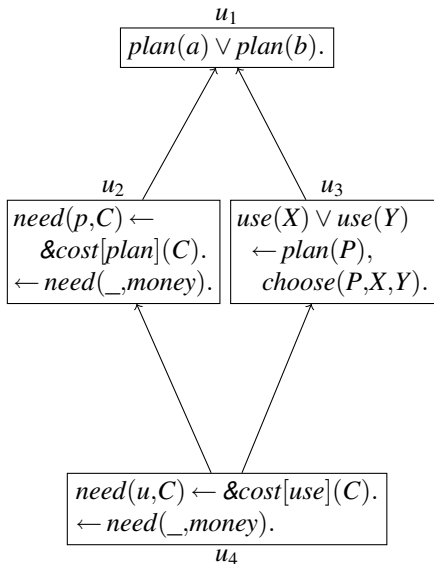


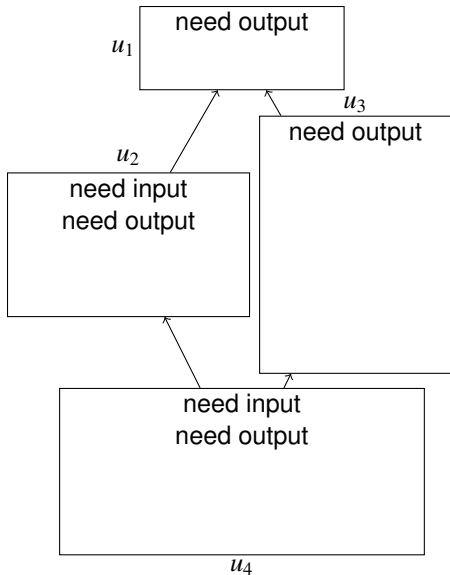
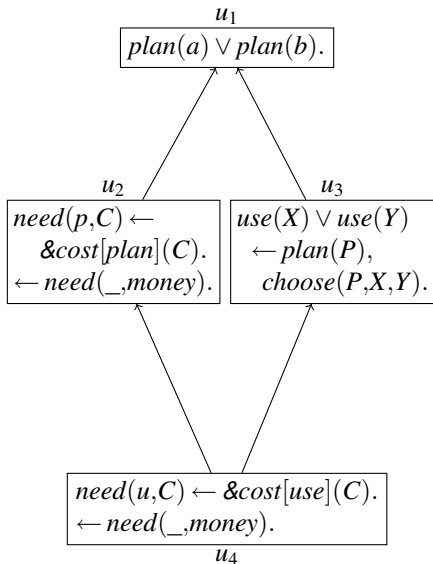
Input and Output models at Evaluation Units:

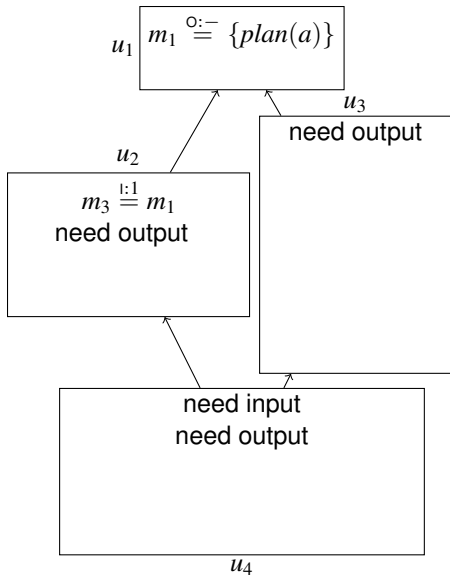
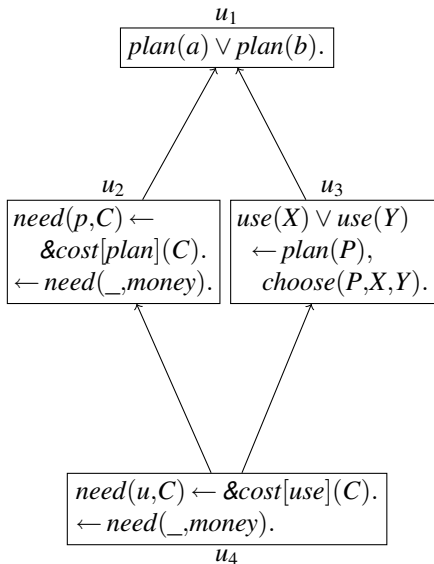
- ▶ input models  $\Leftarrow$  joining output models of predecessor units
- ▶ output models  $\Leftarrow$  evaluating program fragment on input models
- ▶ Principle: **same ancestor** at common ancestor units

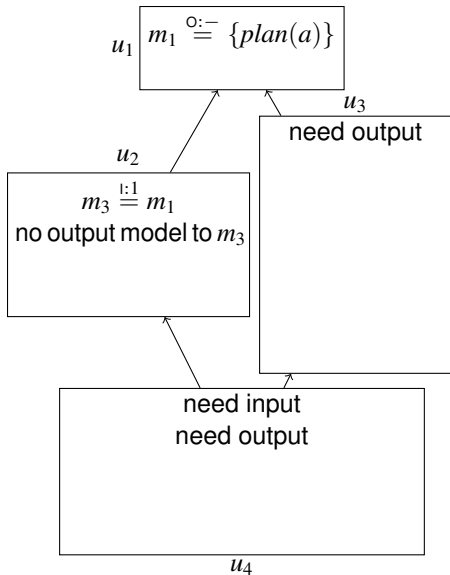
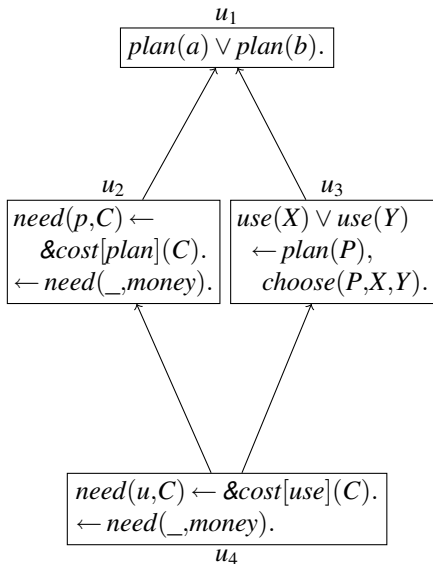
**Central Theorem:**  $I =$  union of connected output models at every unit  
iff  $I =$  answer set of the program

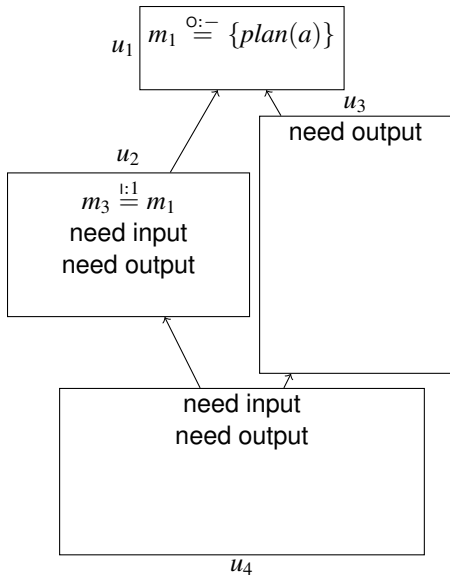
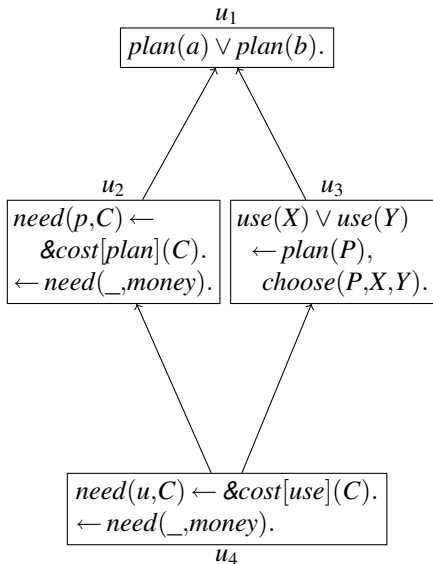


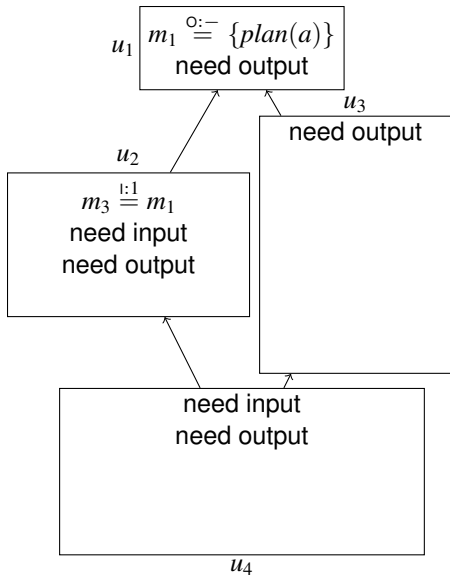
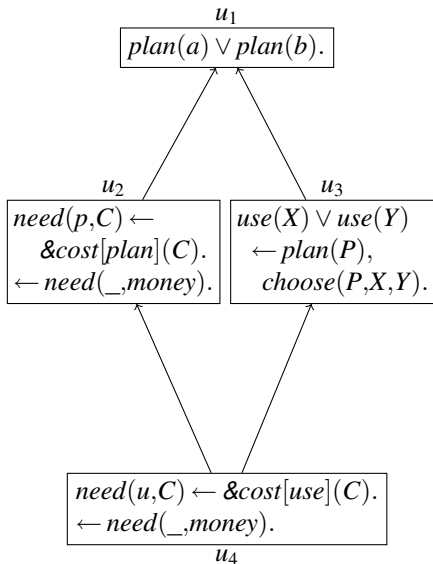


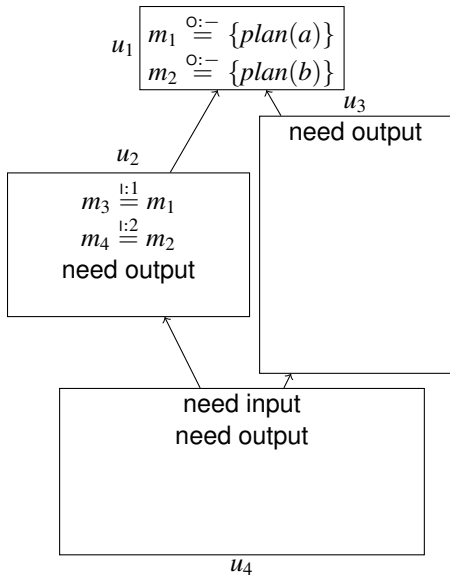
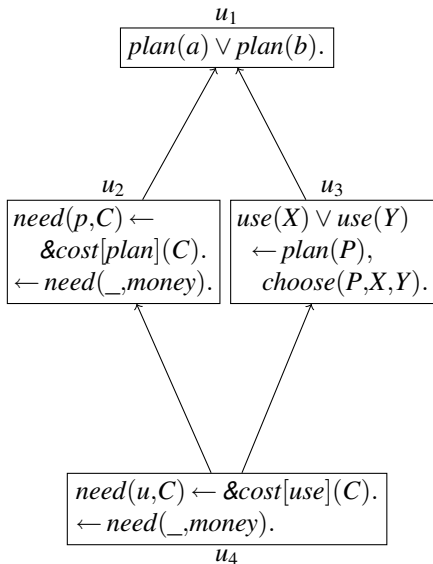


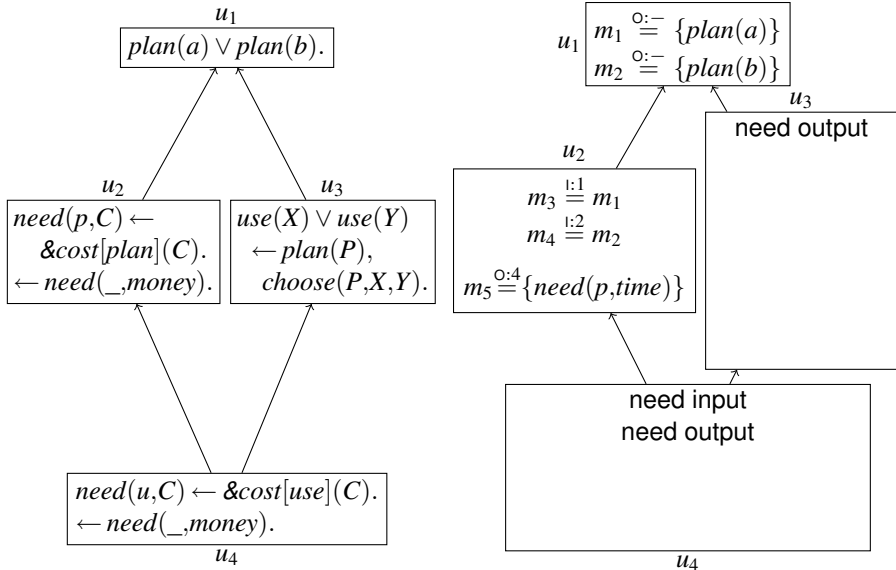


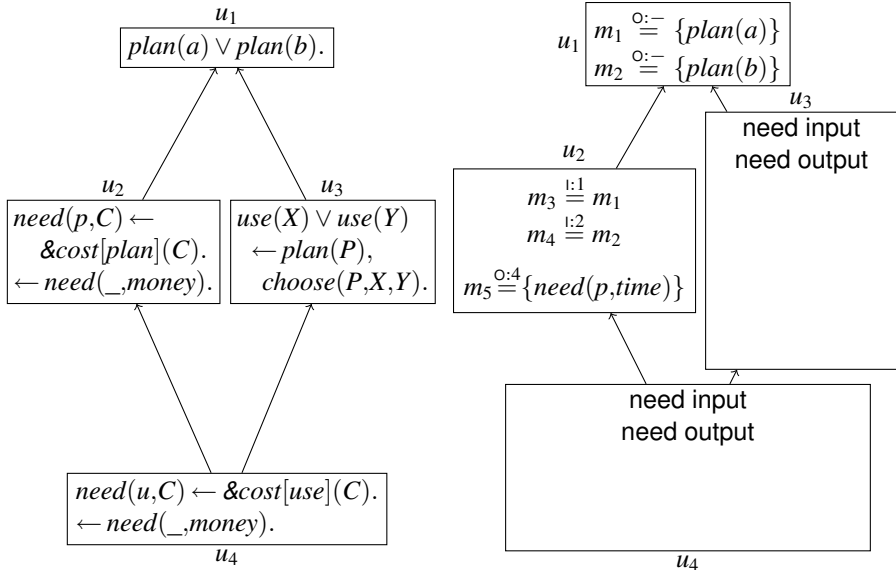


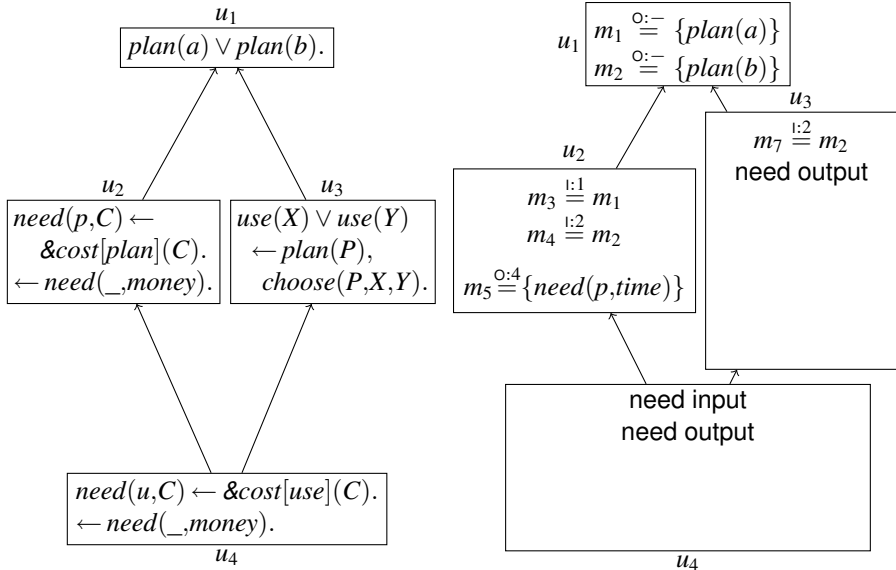


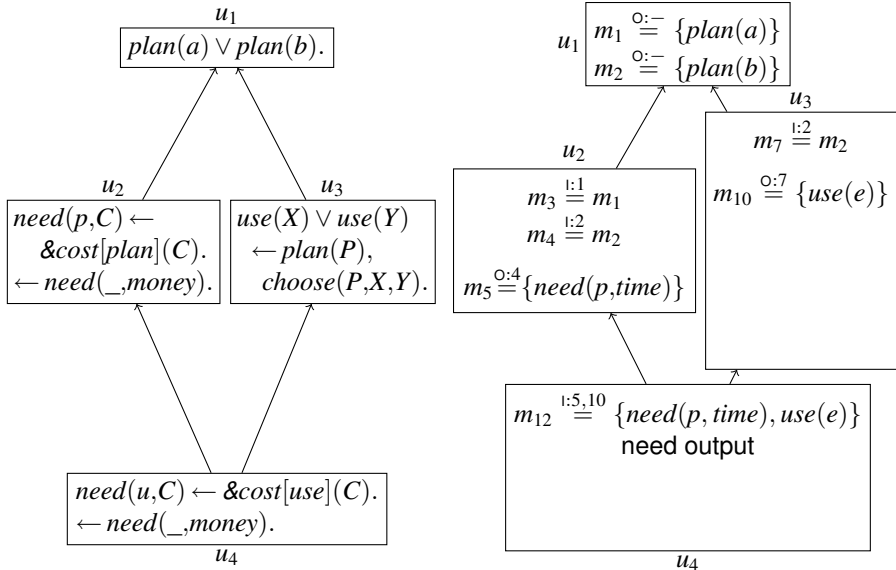


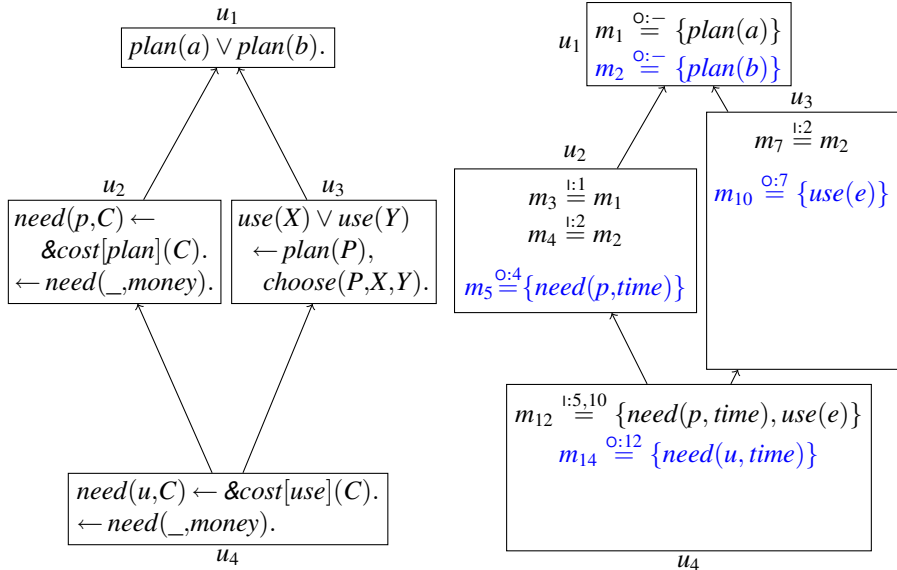


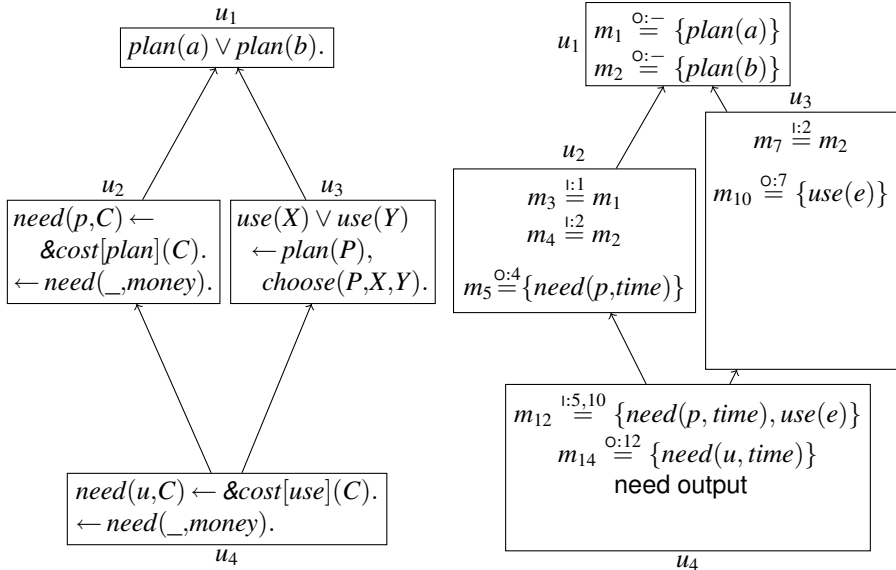


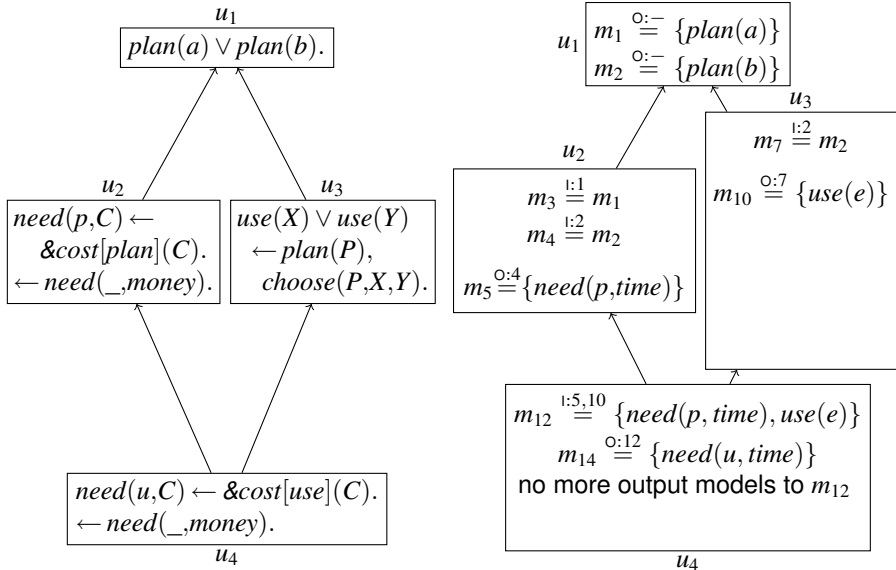


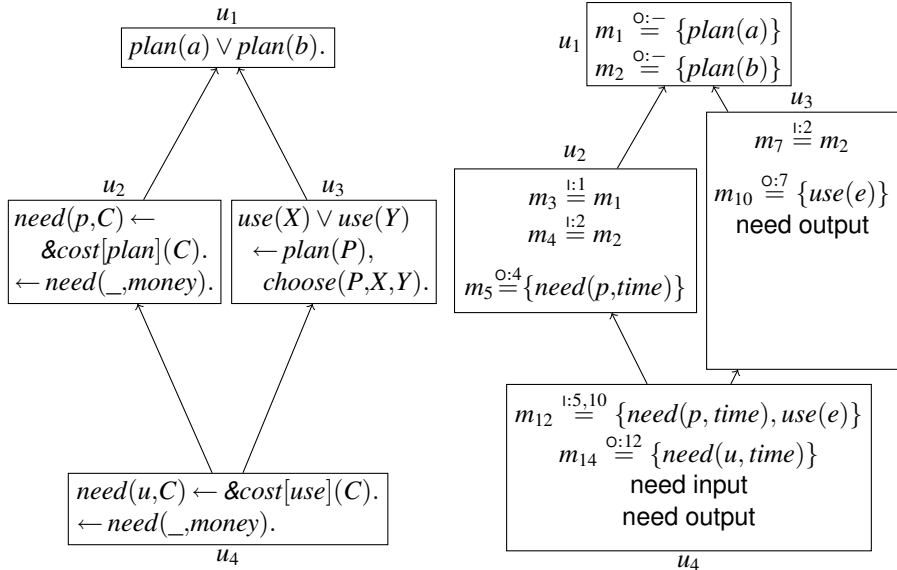


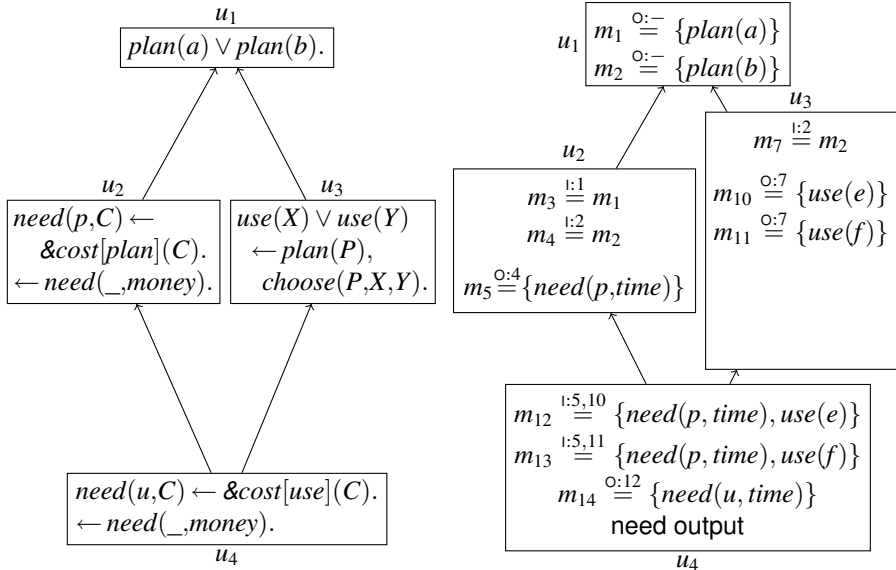


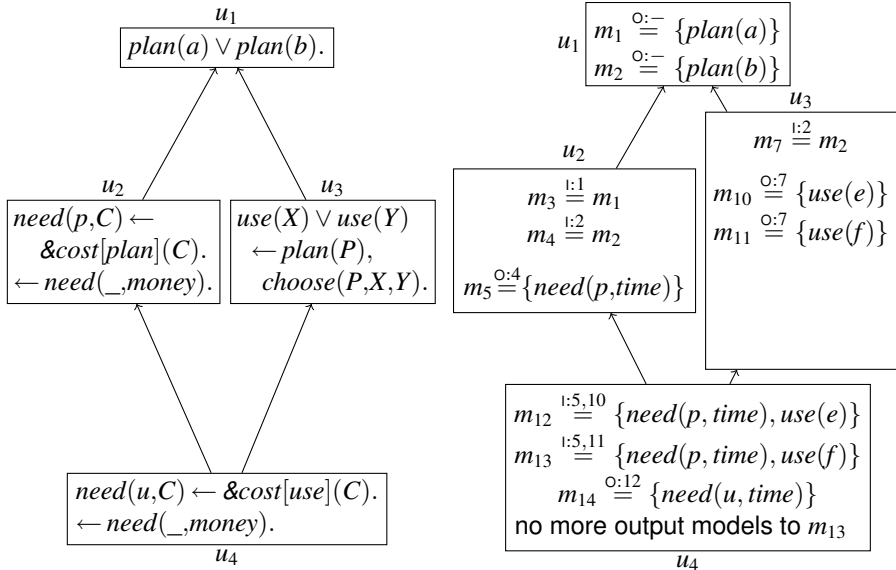


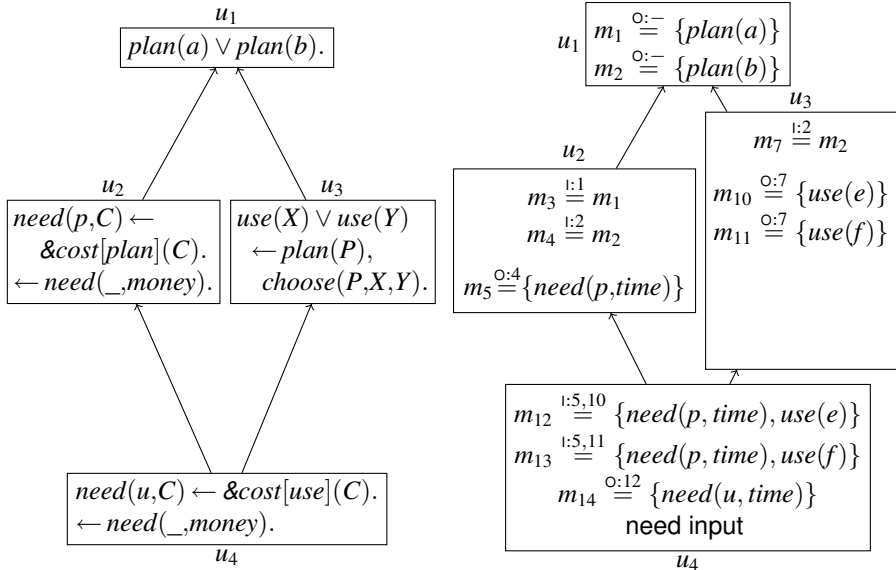


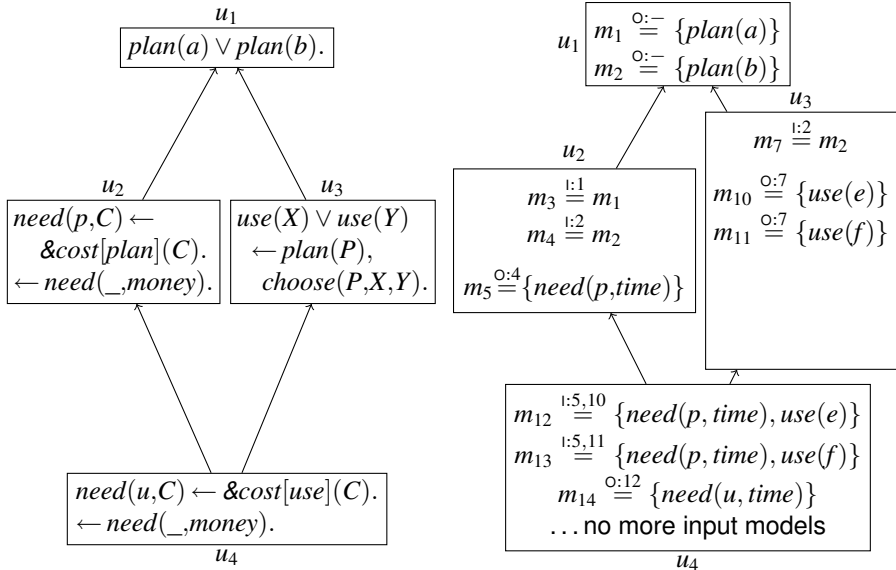


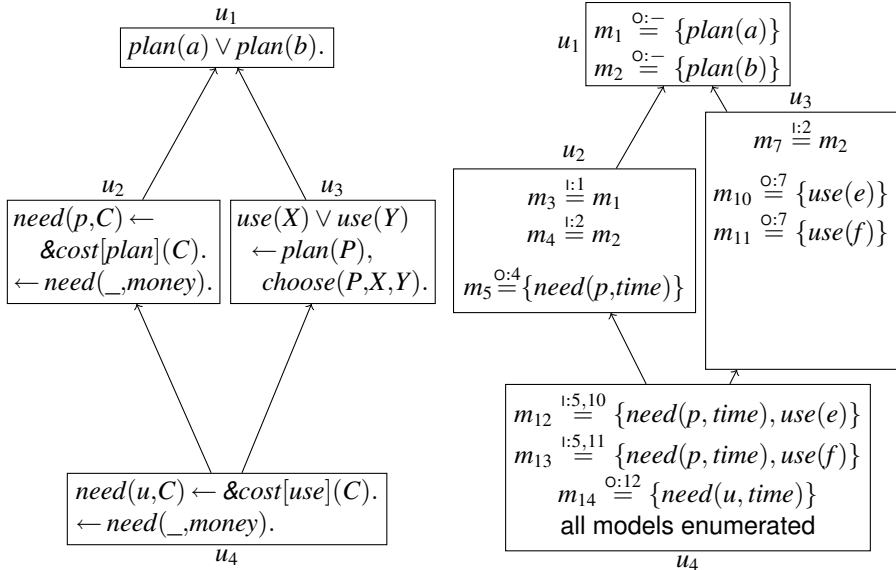


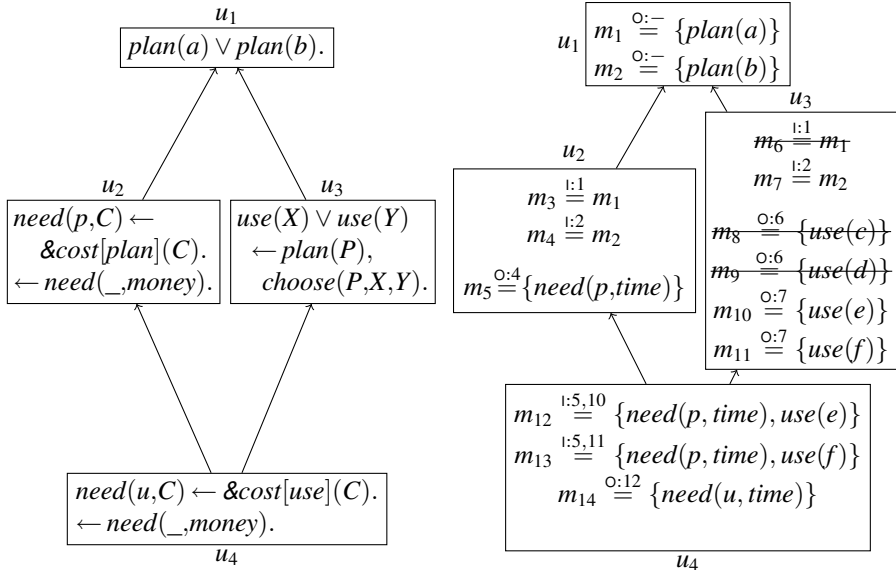














## Acyclic Directed Evaluation Graph

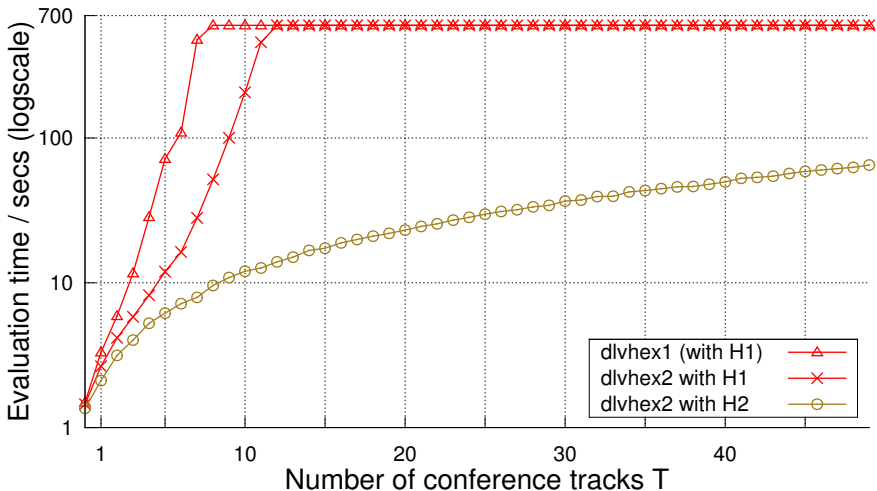
- ⇒ independent program fragments evaluated independently
- ⇒ fewer redundant evaluations
- ⇒ parallel evaluation possible

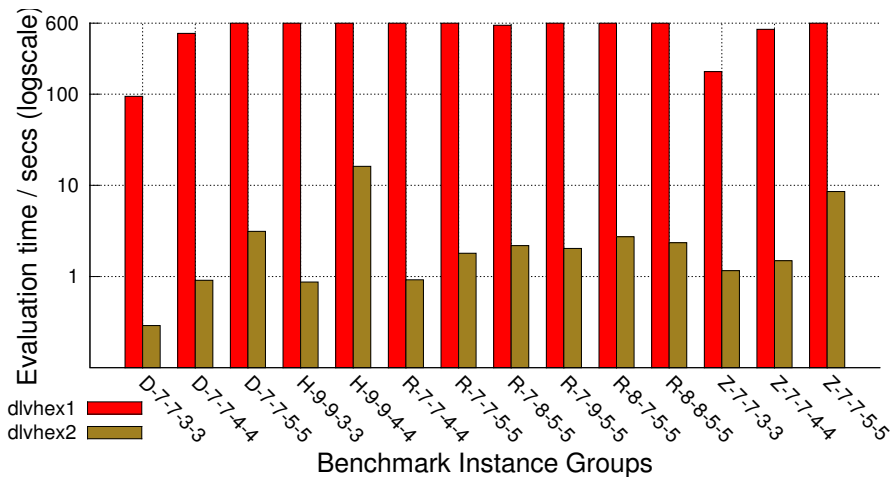
Constraints can be shared between units

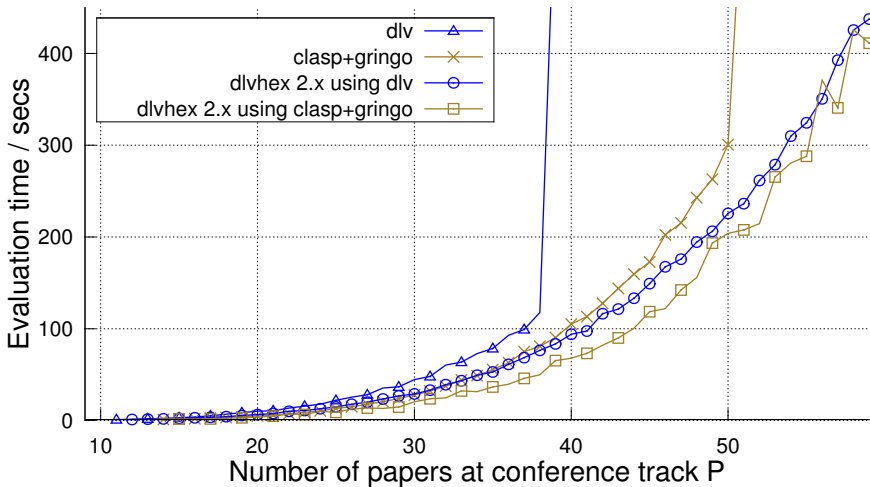
- ⇒ early model elimination

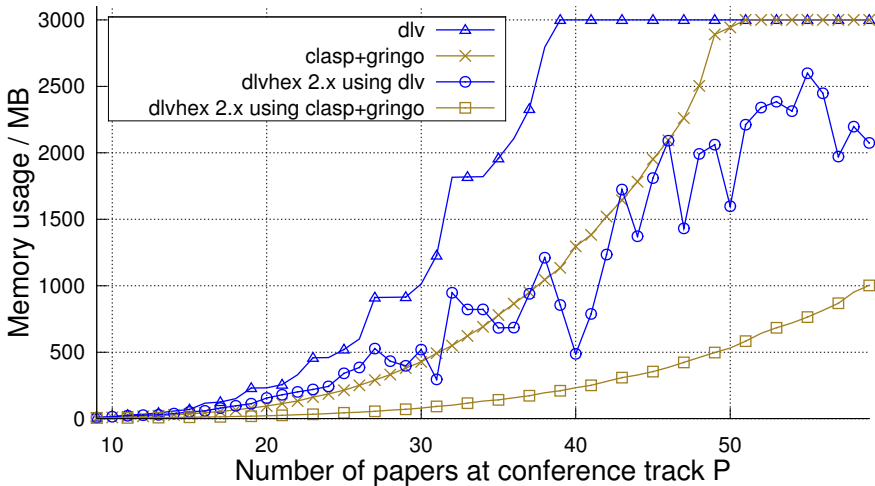


- ▶ Model generation as **streaming mode**
  - ⇒ time-efficient first model computation
  - ⇒ space-efficient model enumeration  
(output answer set → delete part of graph)
  - ⇒ ground and nonground query answering (**new**) for HEX
- ▶ **64bit integer representation** for any entity (space/time-efficiency)
- ▶ **compressed bitset** data structures for interpretations
- ▶ **multiple backends**: dlv and clingo (**new**) integrated











## New HEX evaluation **framework**

- ⇒ simple optimizer heuristics; **exponential speedup** for certain instances
- ⇒ potential for various **future optimization strategies** (parallelization vs. memory usage vs. ... ?)

## New **Implementation** (nearly a full rewrite)

- ⇒ try it and tell us about your experiences/problems (not yet released, but branch is fully functional)
- ⇒ Instructions for building and detailed experimental results: click `Experiments` on the `dlvhex` homepage  
`http://www.kr.tuwien.ac.at/research/systems/dlvhex/`



- ▶ 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.
- ▶ M. Gelfond and V. Lifschitz. Classical Negation in Logic Programs and Disjunctive Databases. *Next Generation Computing*, 9(3–4):365–386, 1991.