FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

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

LPNMR – May 18, 2011

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

Outline

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

Answer Set Programming [Gelfond and Lifschitz, 1991]

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

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

Program P is a set of rules.

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

Interpretation $I \subseteq HB_P$.

- $I \models grnd(P)$ 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.

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

Example Program (IDB):

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

External Atom:

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

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

Example Program (IDB):

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

External Atom:

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

HEX **Example**

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

Example Program (IDB):

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

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.

HEX **Example**

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

Example Program (IDB):

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

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 . . .

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

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

HEX **Example**

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

Example Program (IDB):

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

Answer Set Candidates (Guesses, without EDB):

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

HEX Example (Answer Sets)

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

Example Program (IDB):

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

Answer Set Candidates (Guesses, without EDB):

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

HEX **evaluation (old strategy)**

(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)

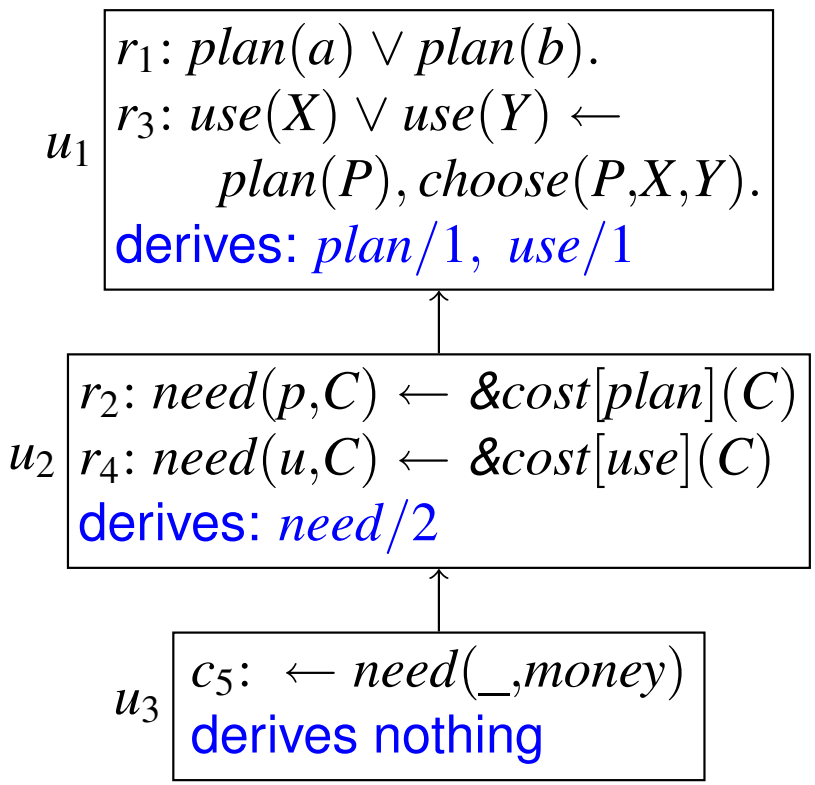
HEX evaluation (old strategy)

(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)

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



HEX model building (old strategy)

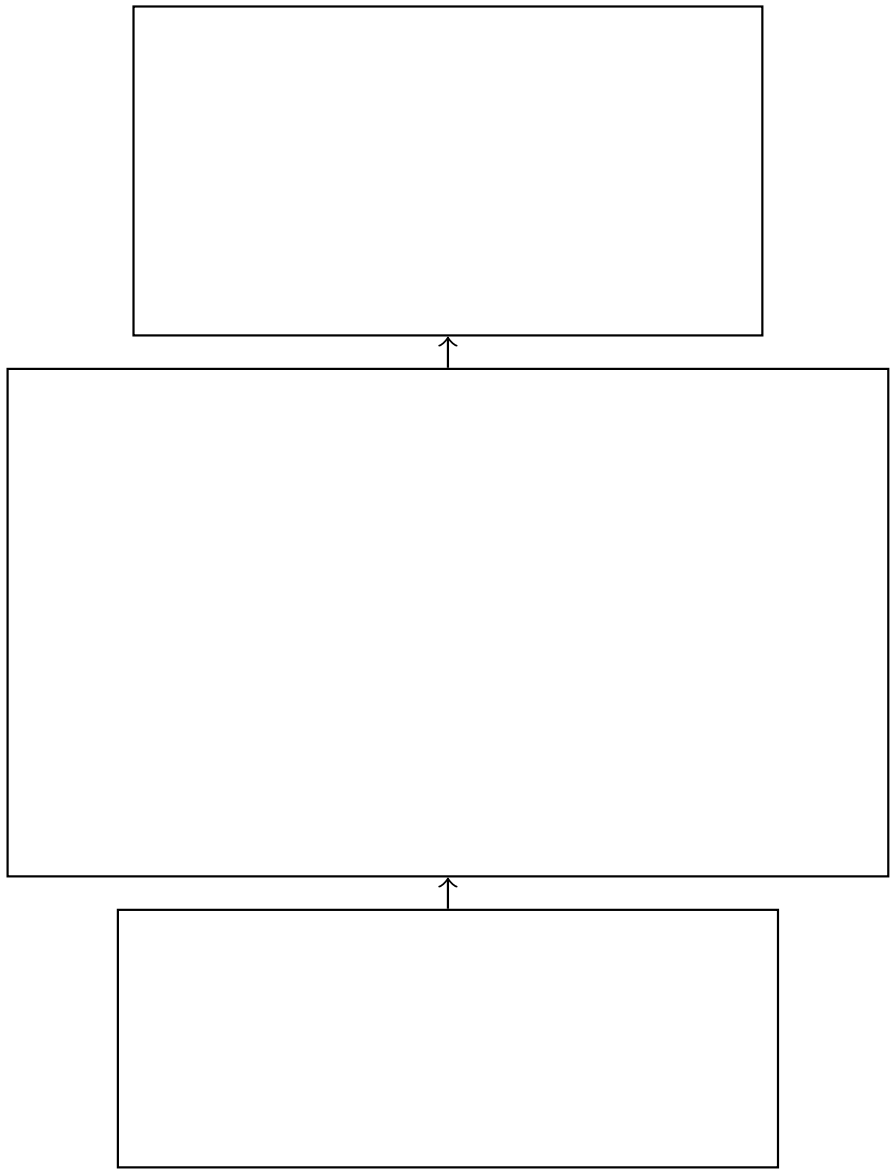
"Evaluation Graph"

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

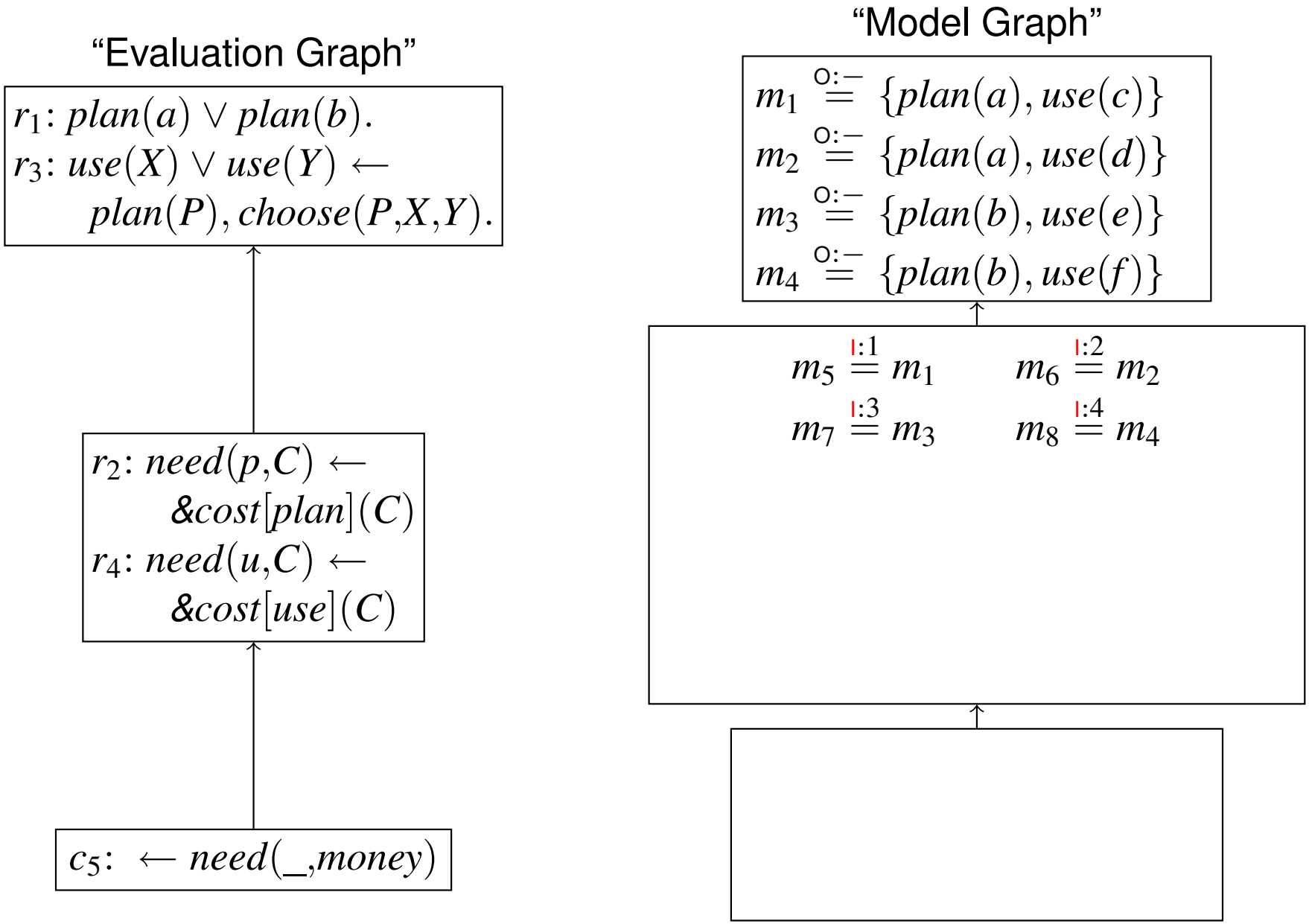
r_2: $need(p,C) \leftarrow$
$\&cost[plan](C)$
r_4: $need(u,C) \leftarrow$
$\&cost[use](C)$

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

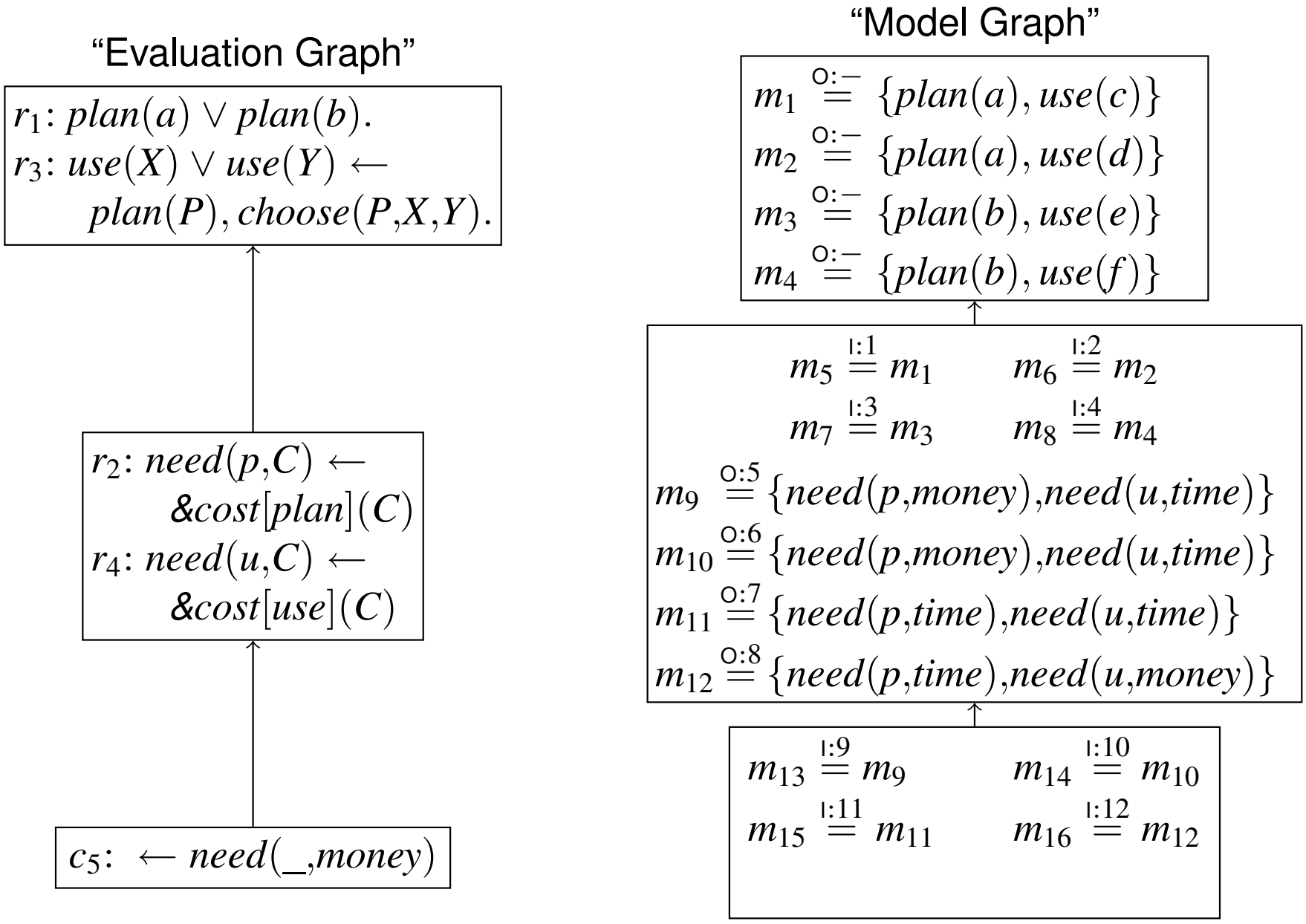
"Model Graph"



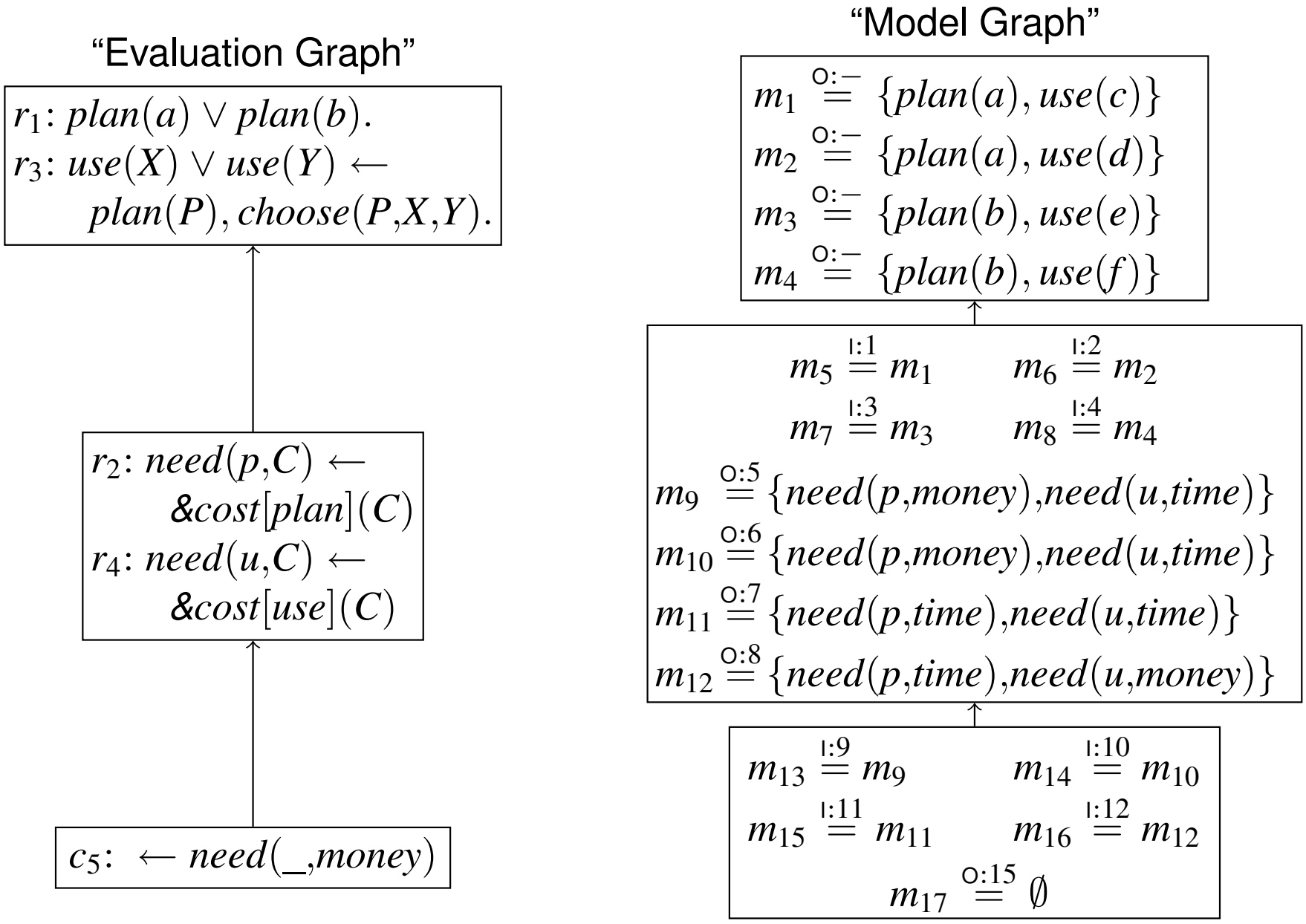
HEX model building (old strategy)



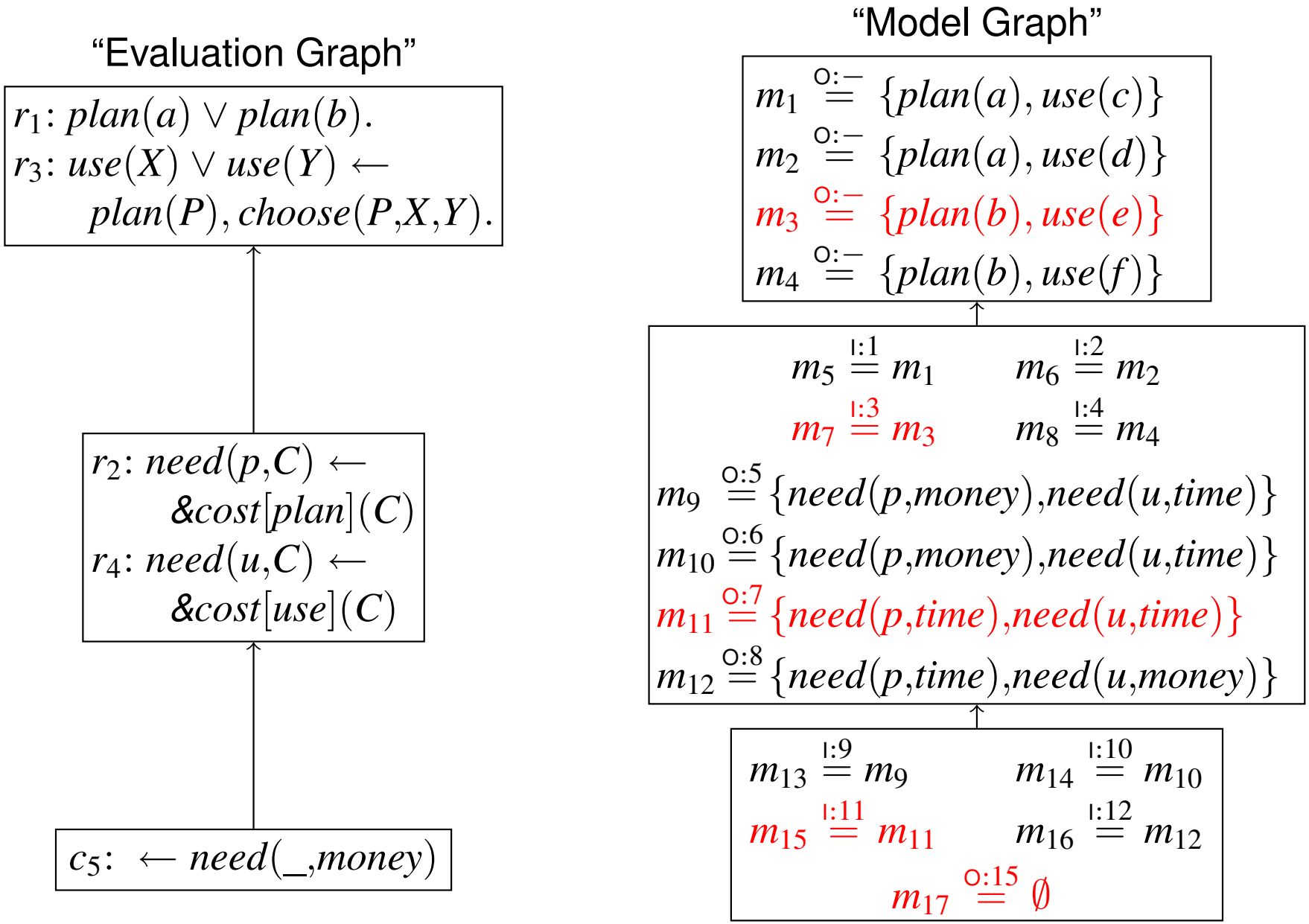
HEX model building (old strategy)



HEX model building (old strategy)



HEX model building (old strategy)

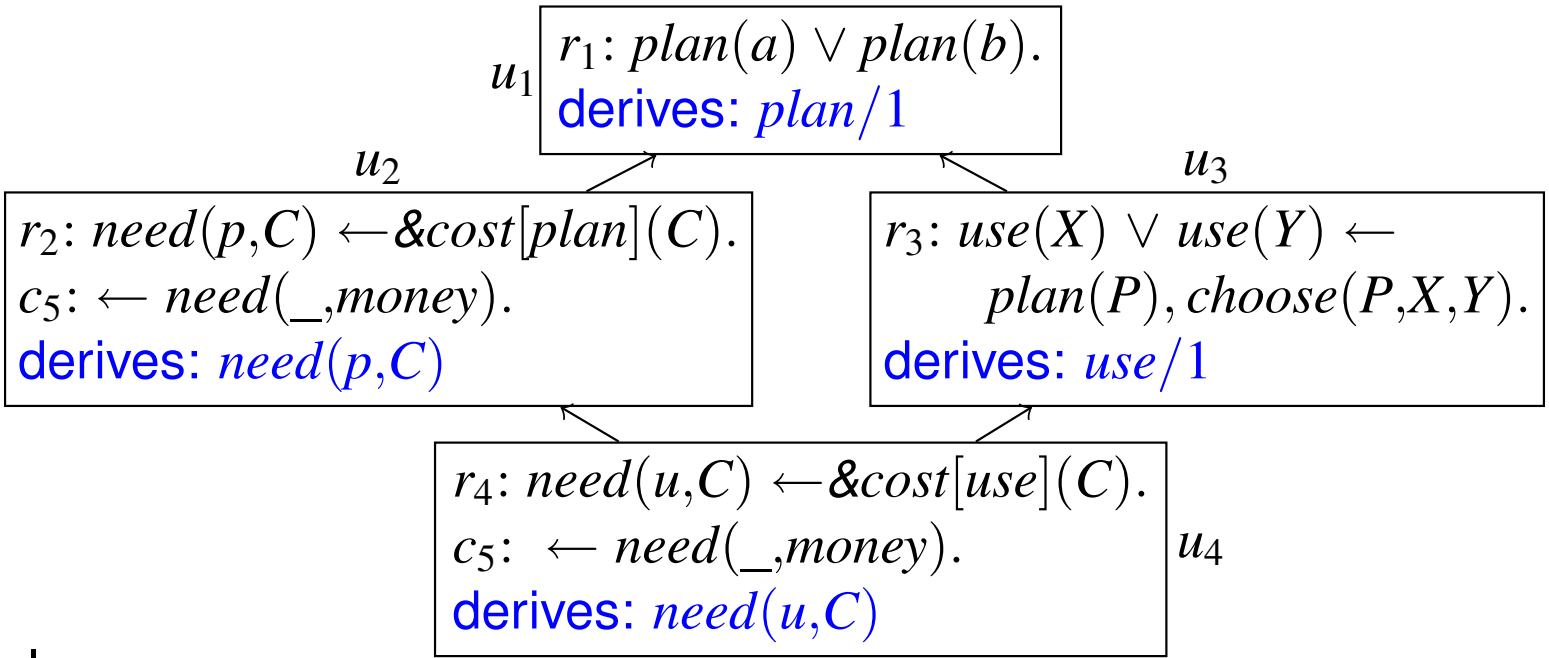


Observations

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

$\Rightarrow$ **Many improvements possible!**

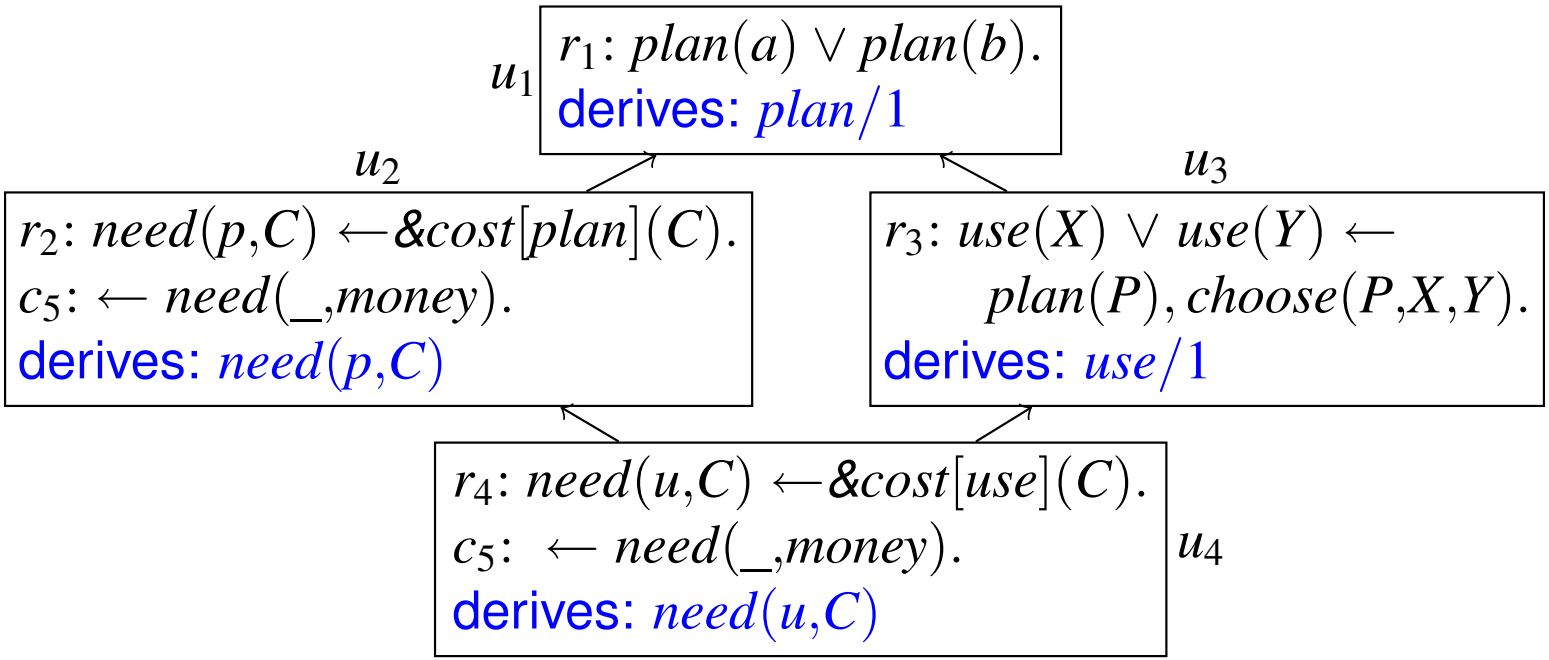
New Strategy Evaluation Graph



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

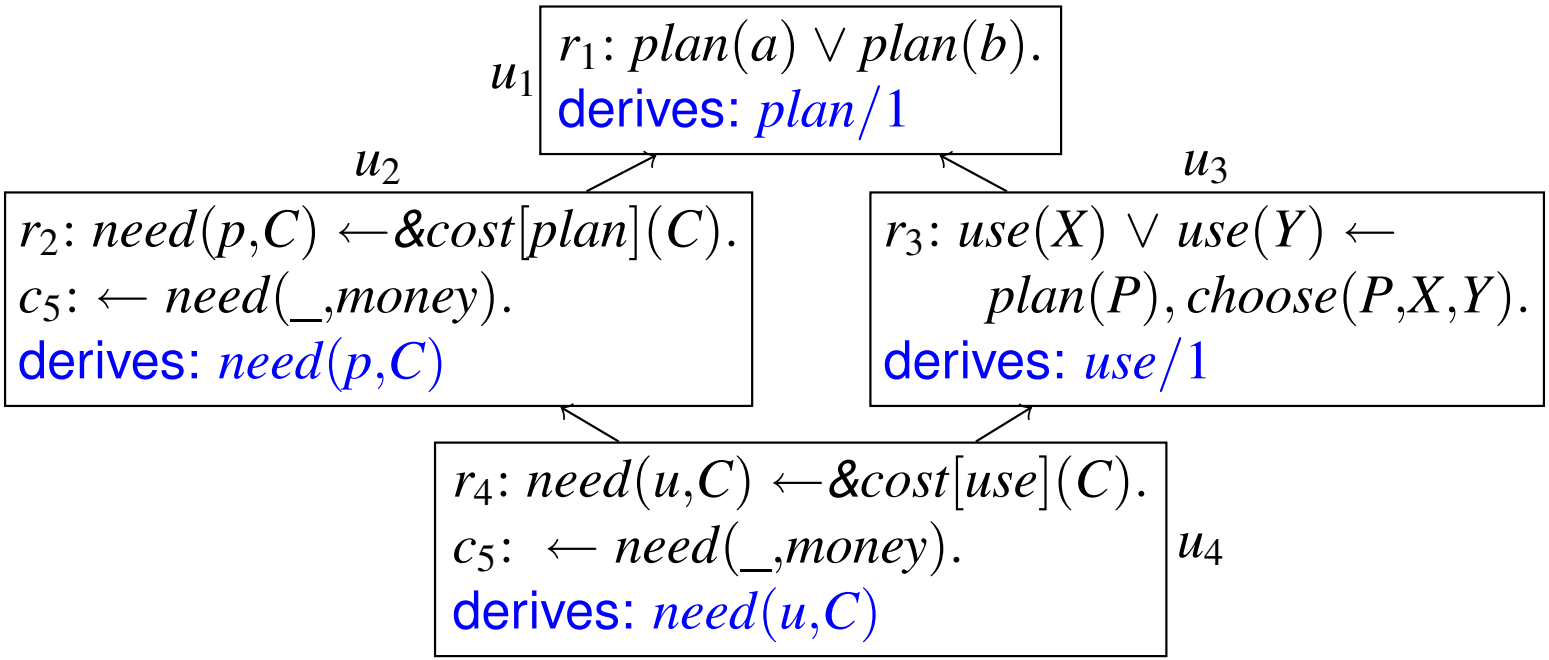
New Strategy Model Building



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

New Strategy Model Building

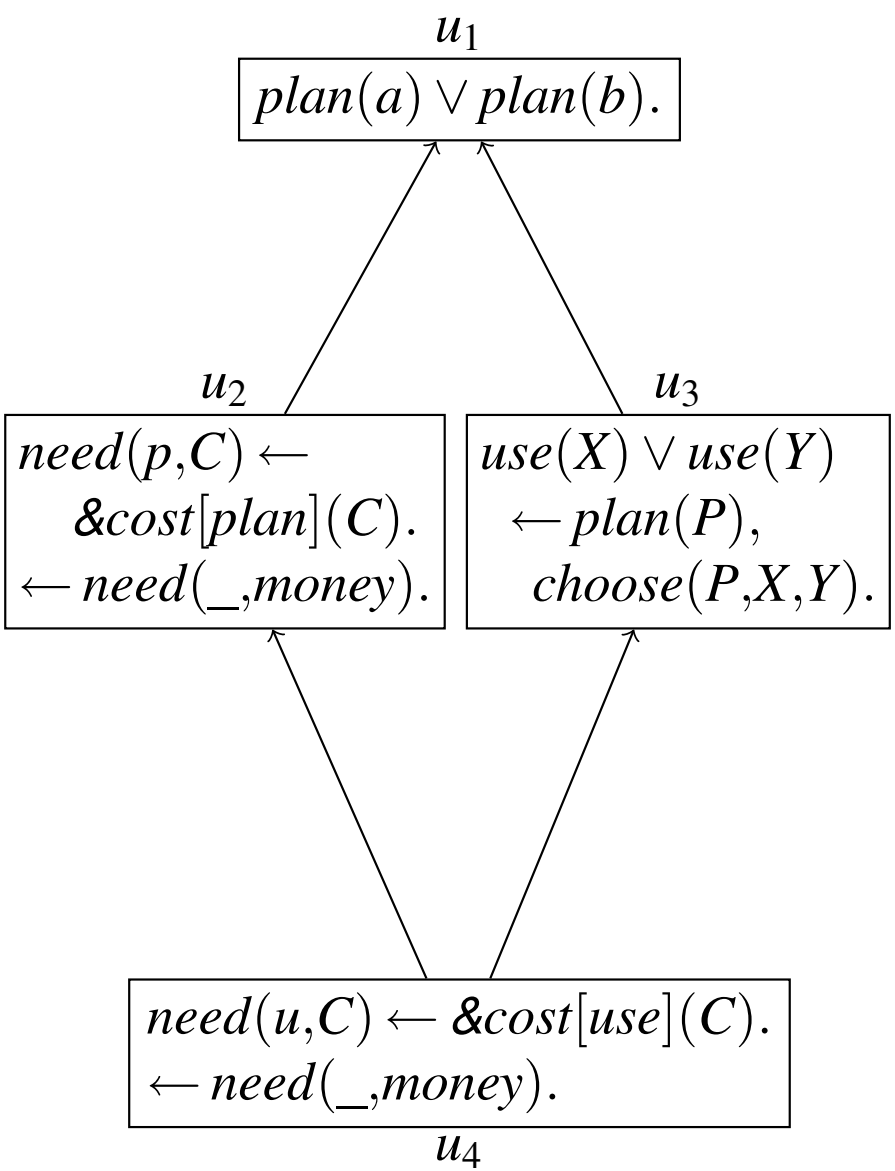


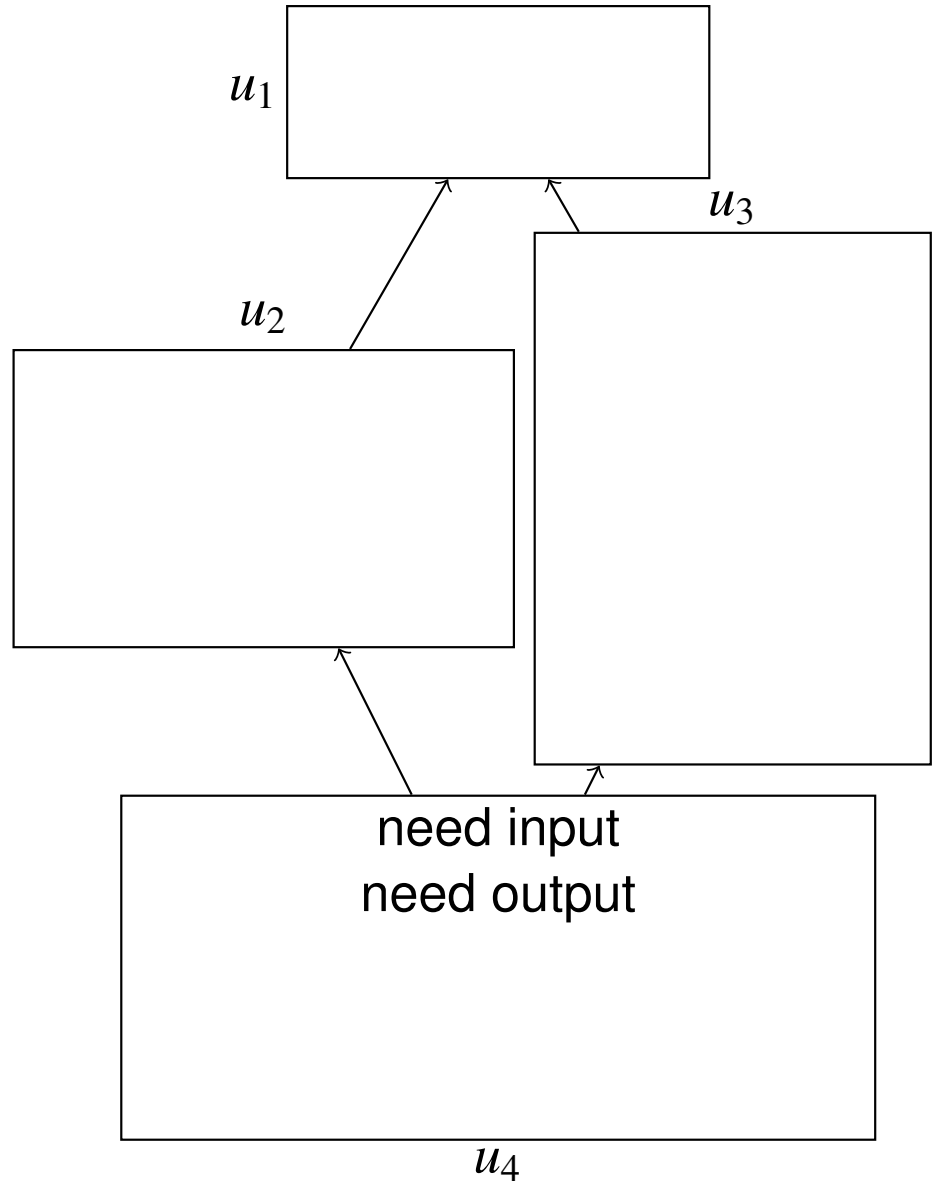
Input and Output models at Evaluation Units:

- 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

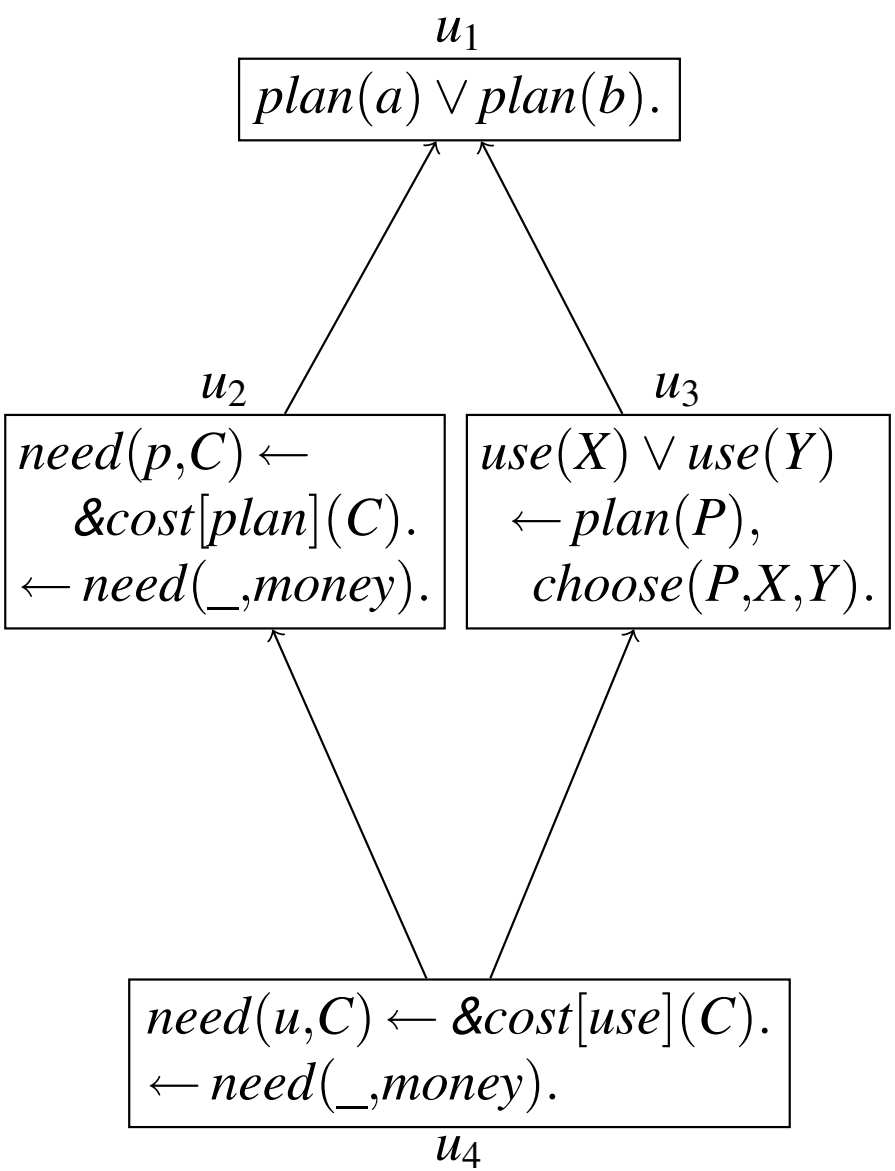
Central Theorem: I = union of connected output models at every unit
iff I = answer set of the program

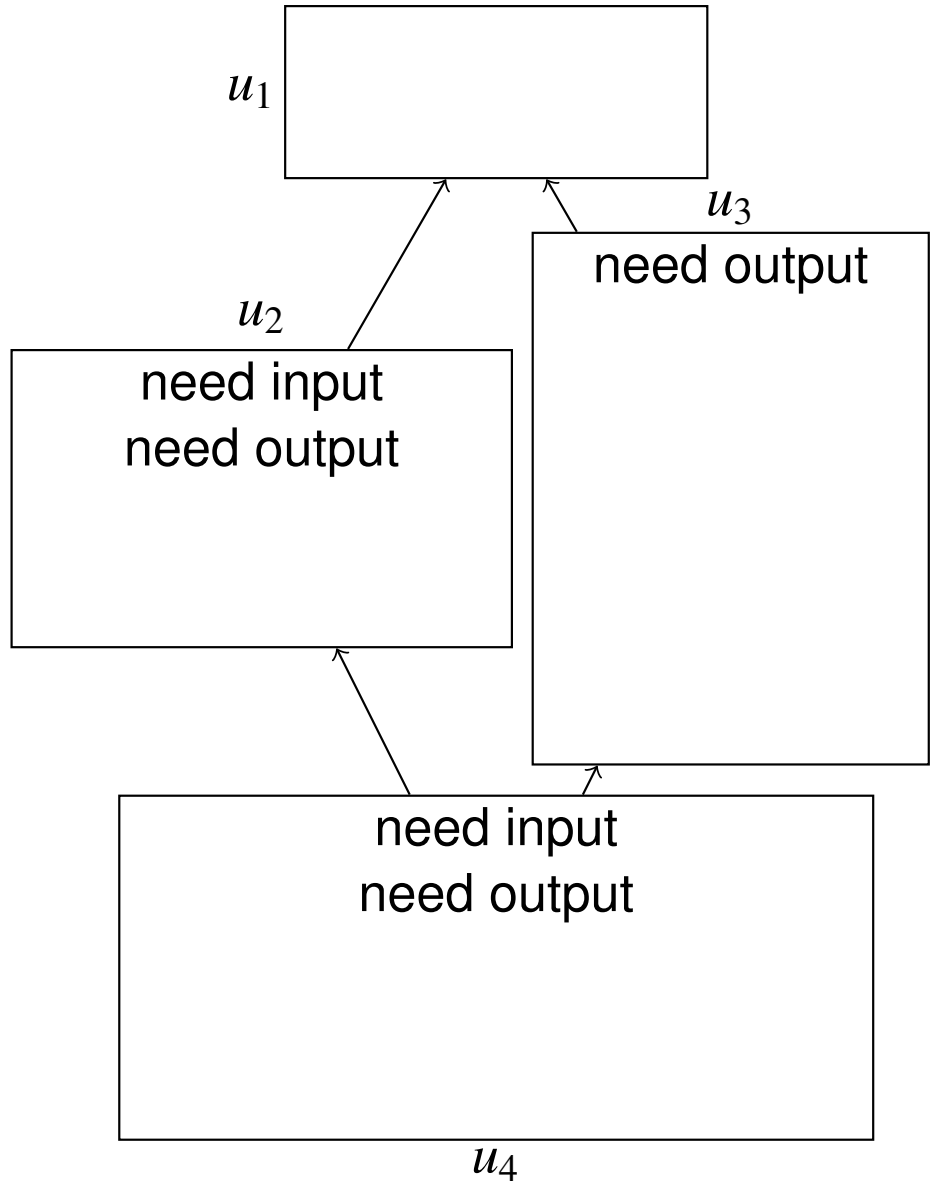
New Strategy Model Building



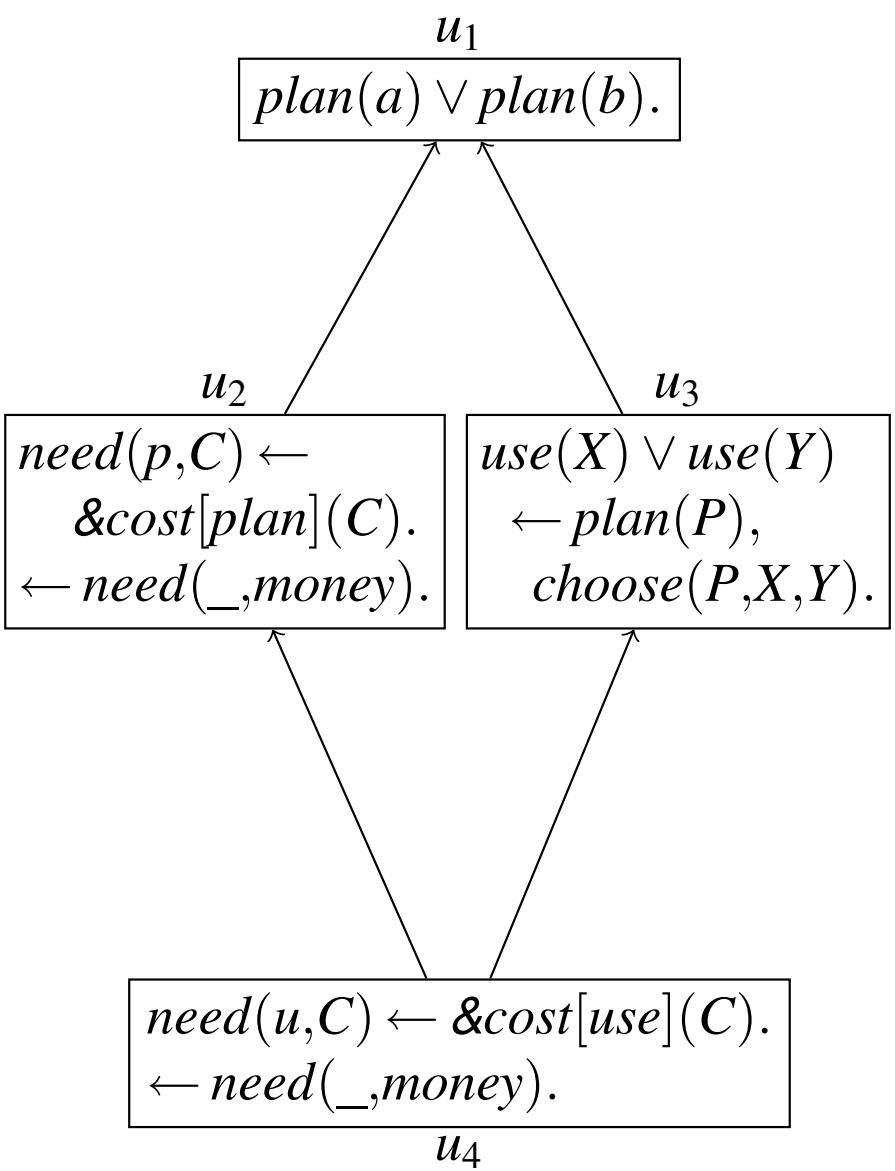


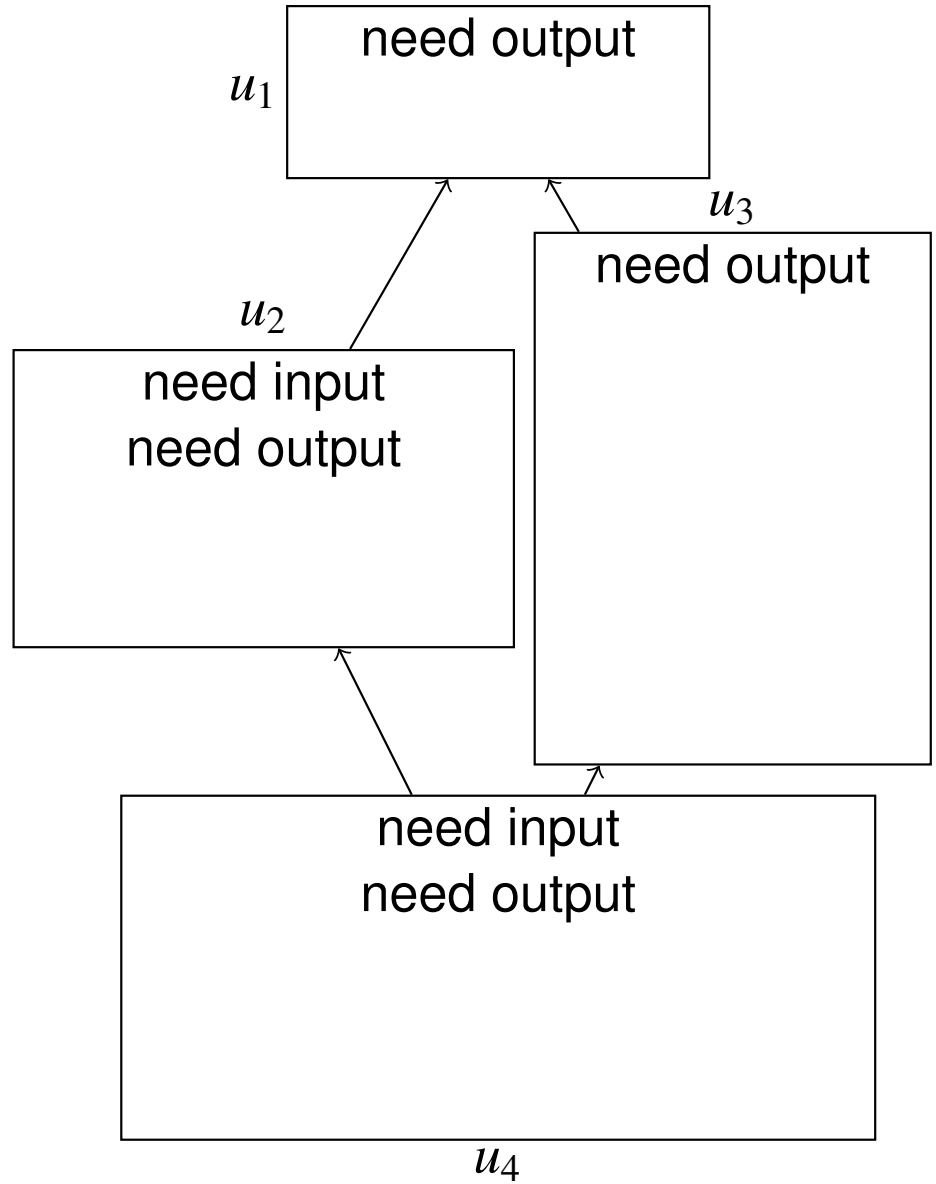
New Strategy Model Building



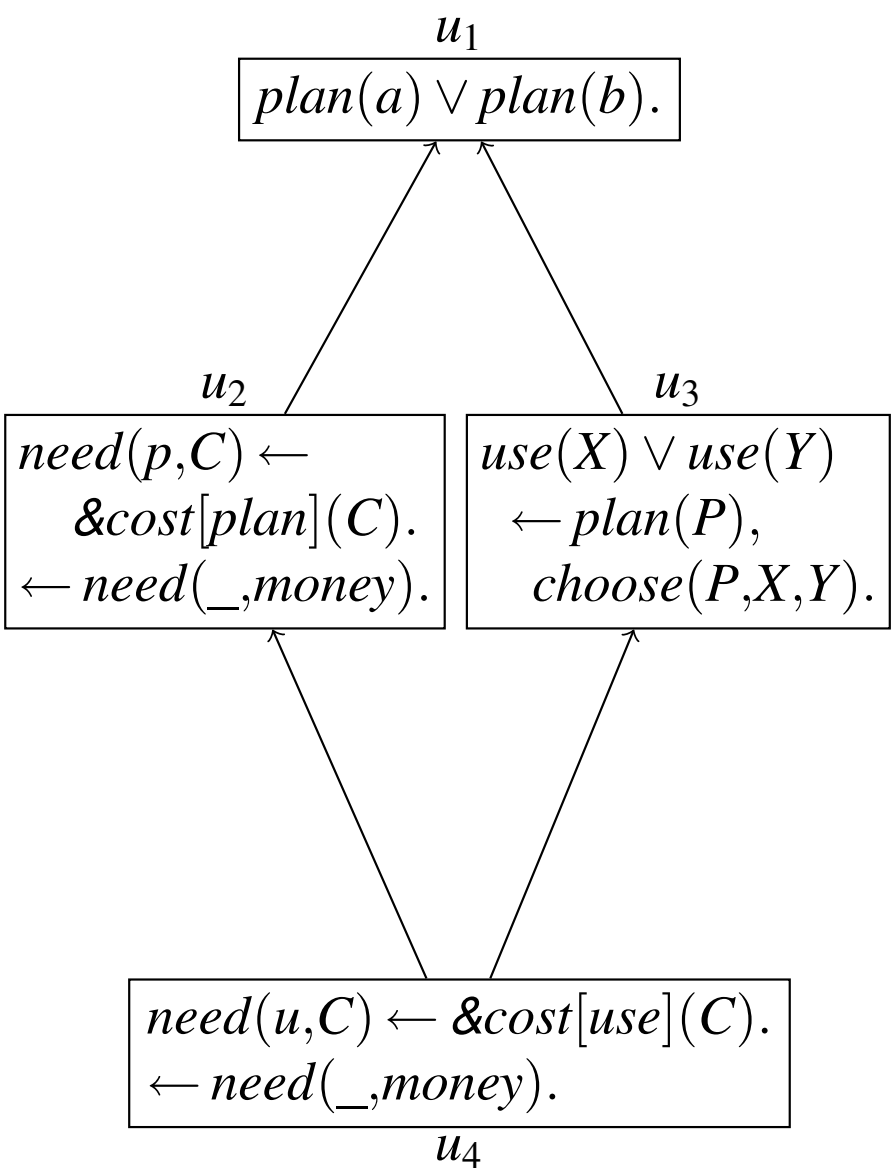


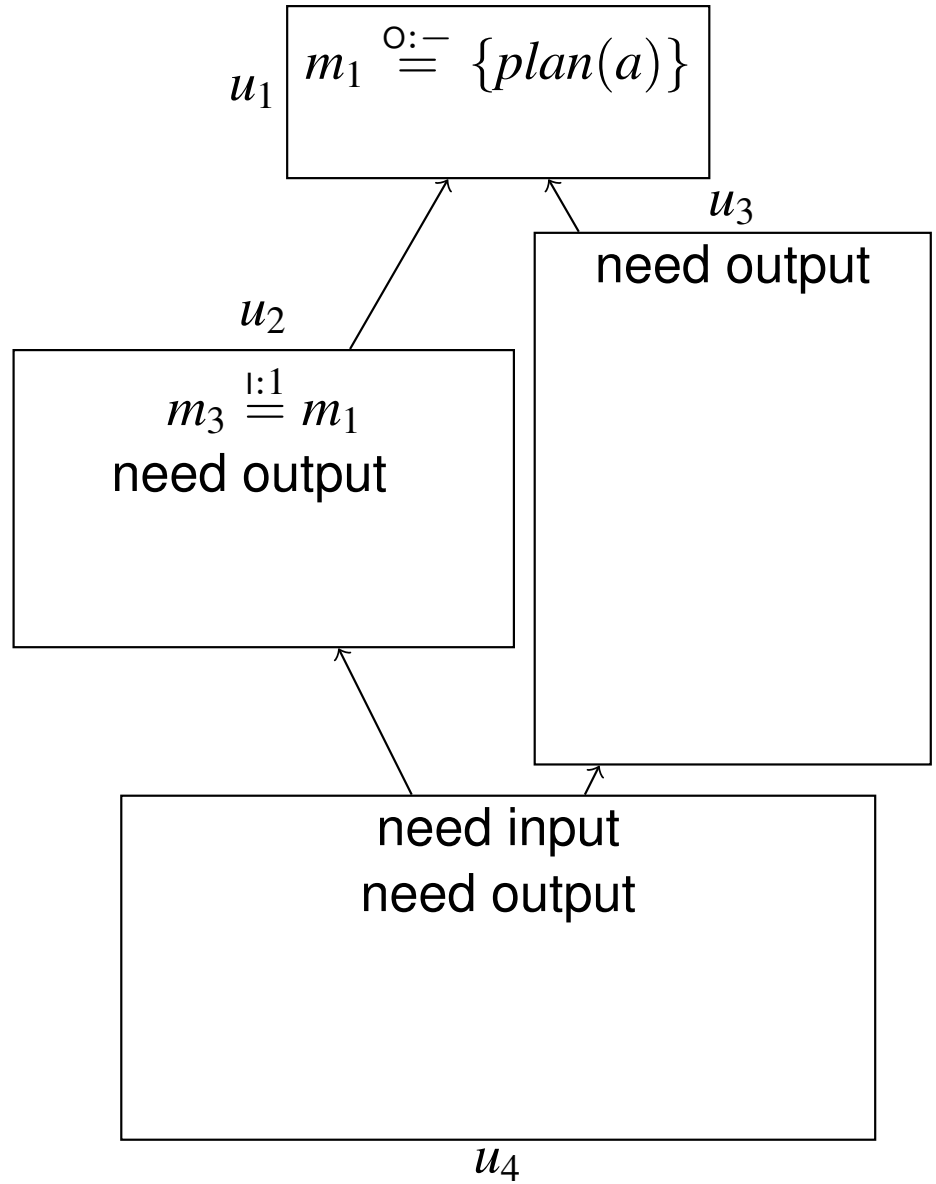
New Strategy Model Building



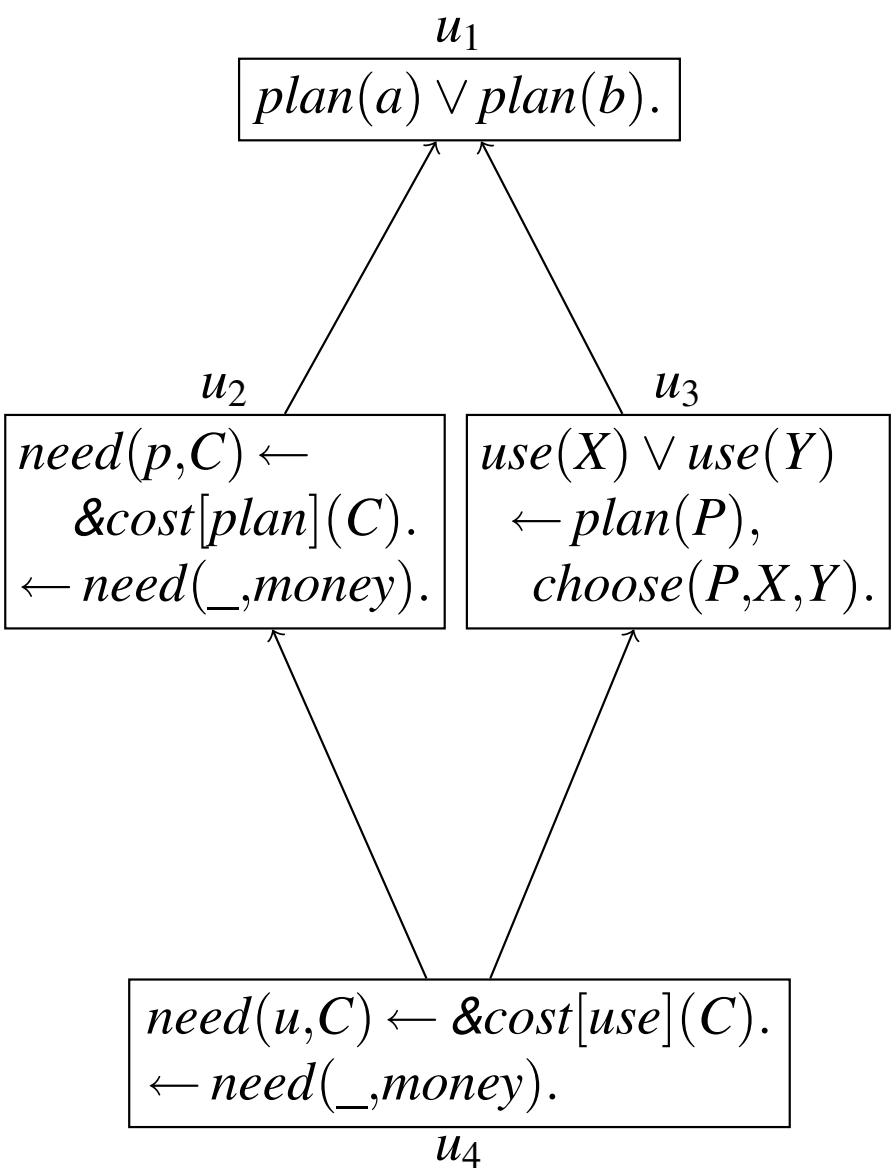


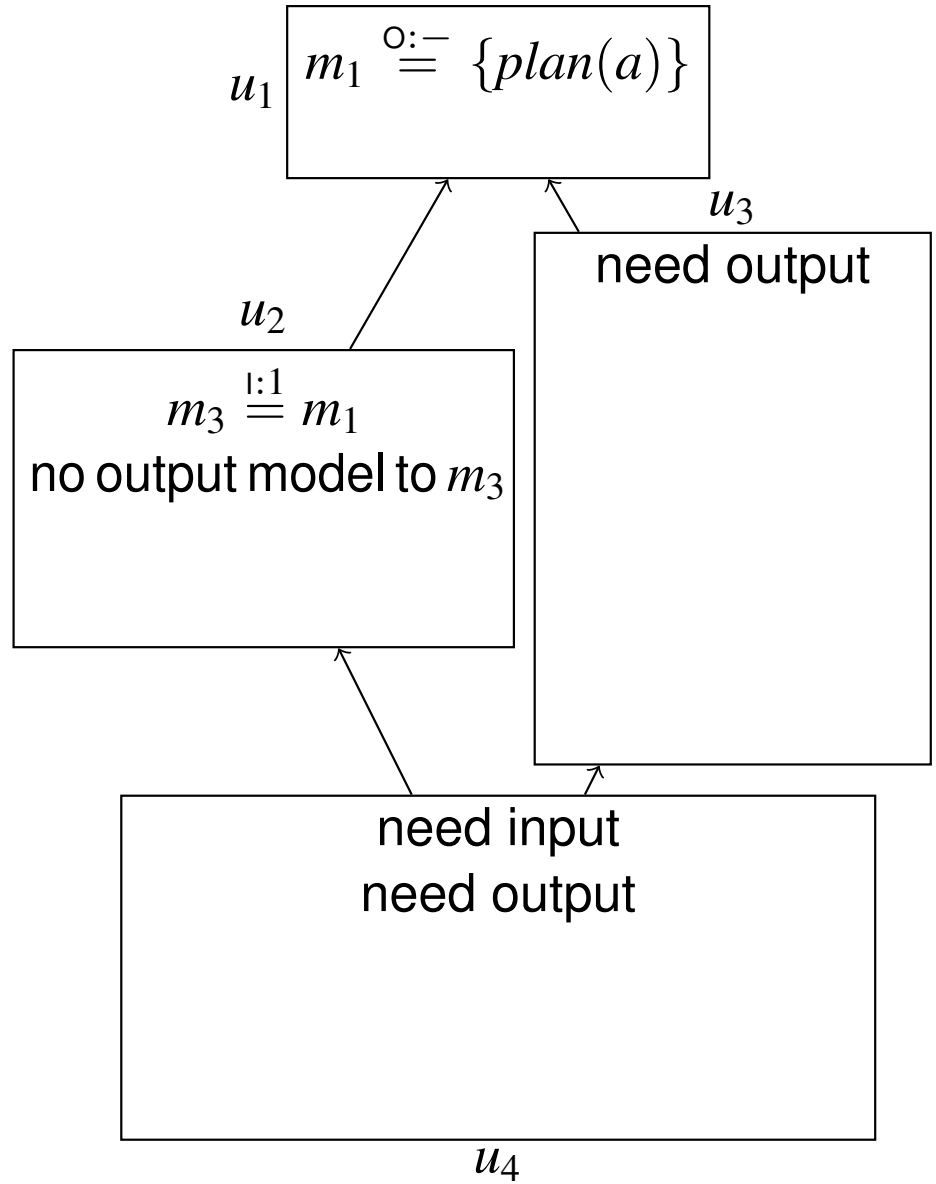
New Strategy Model Building



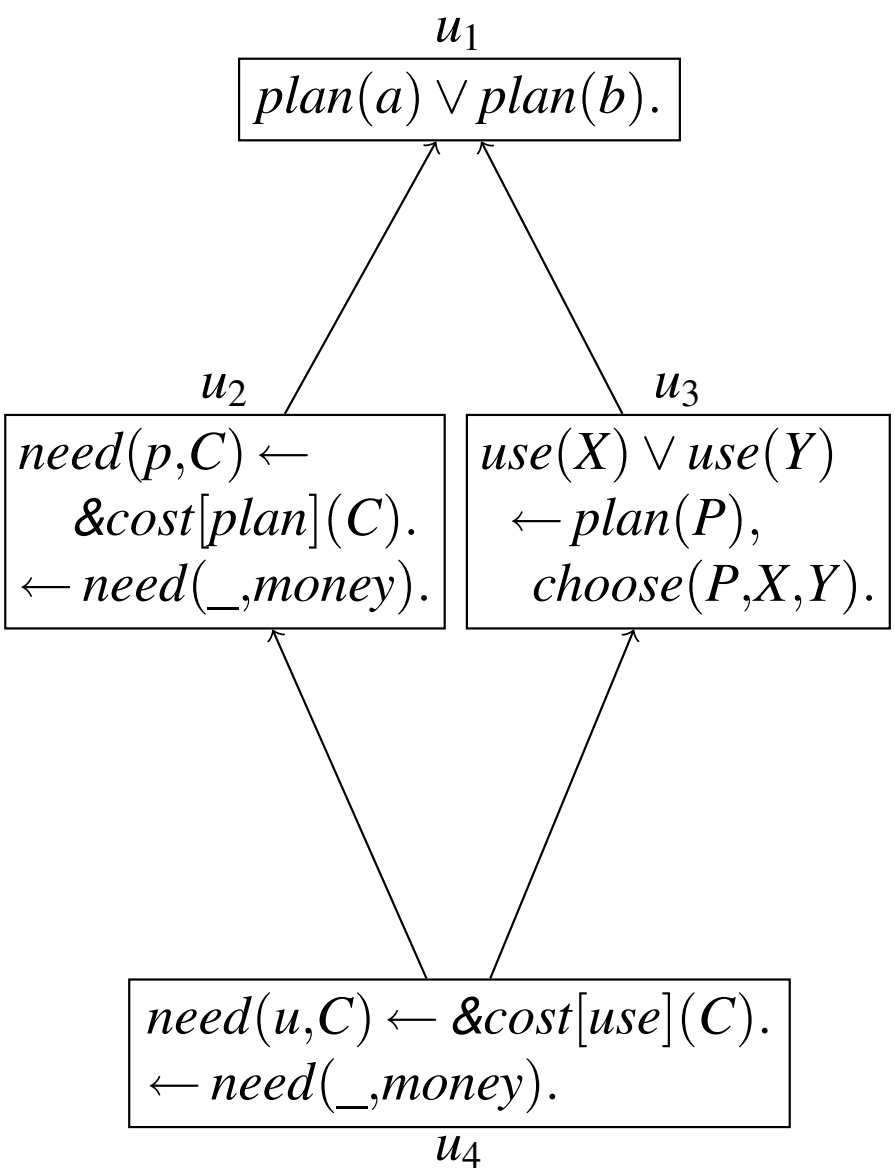


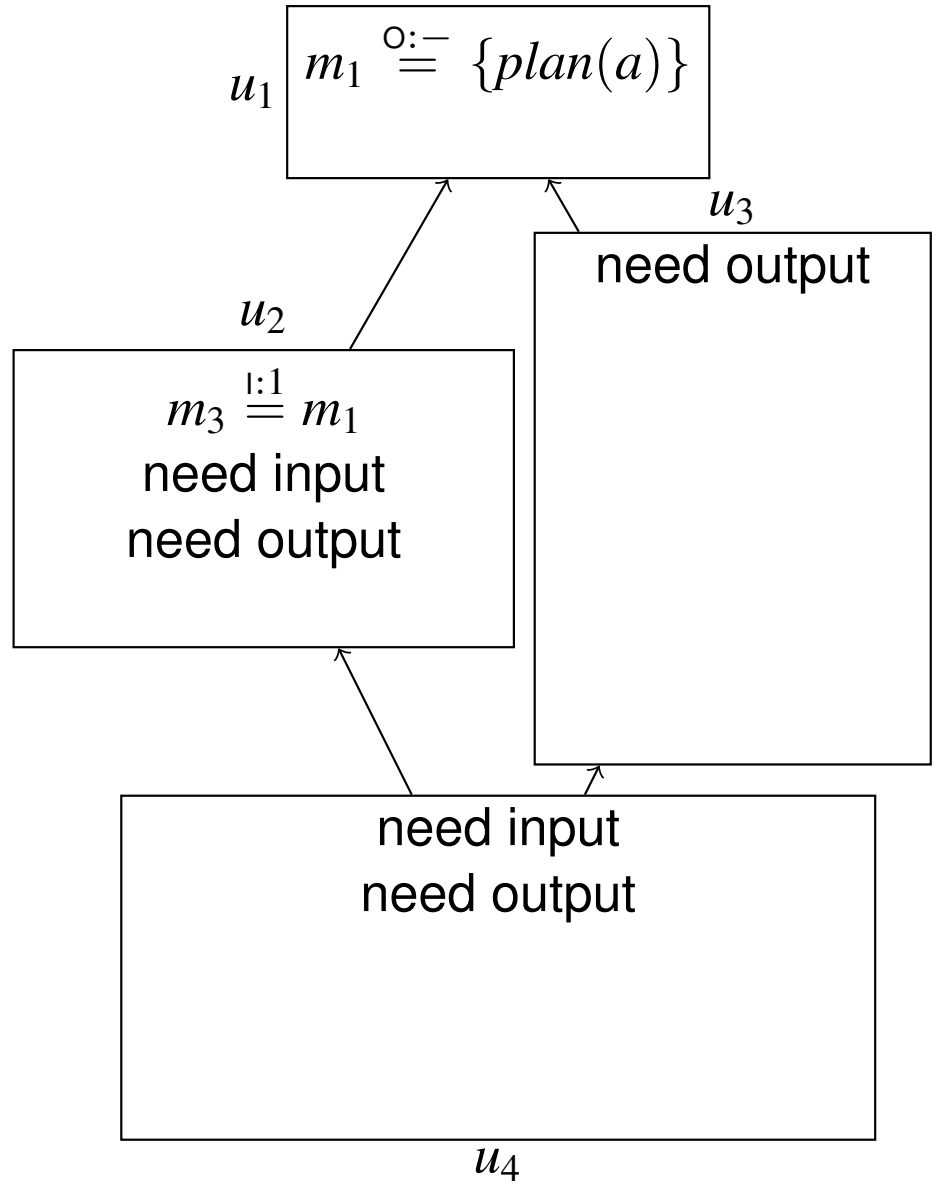
New Strategy Model Building



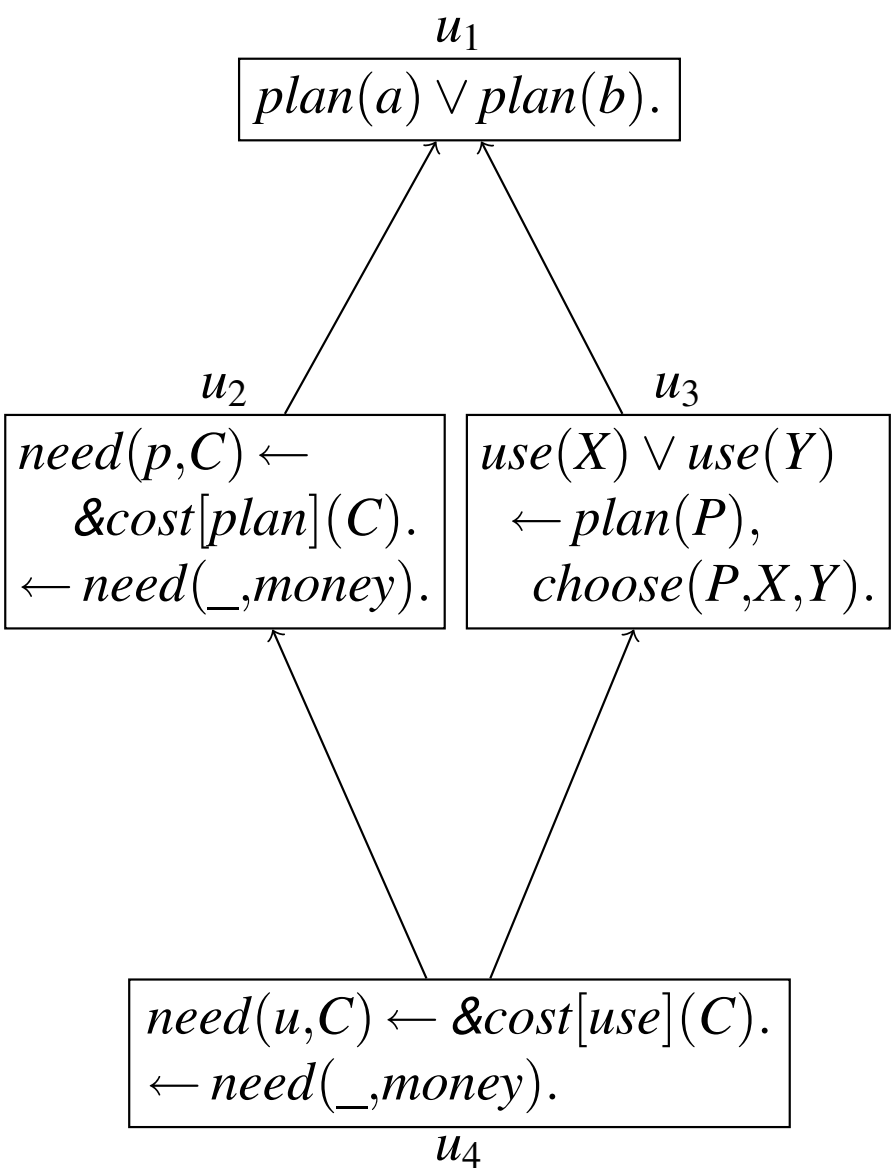


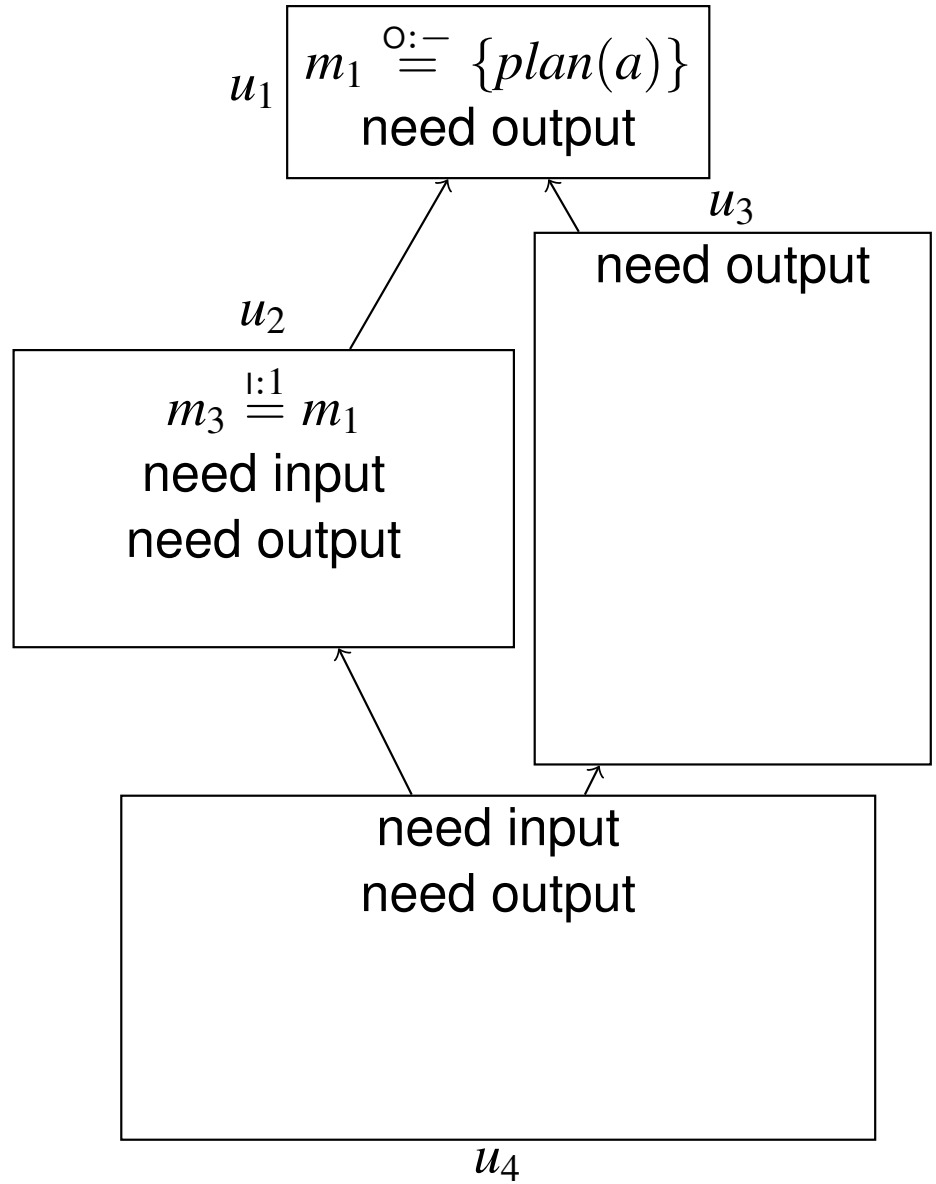
New Strategy Model Building



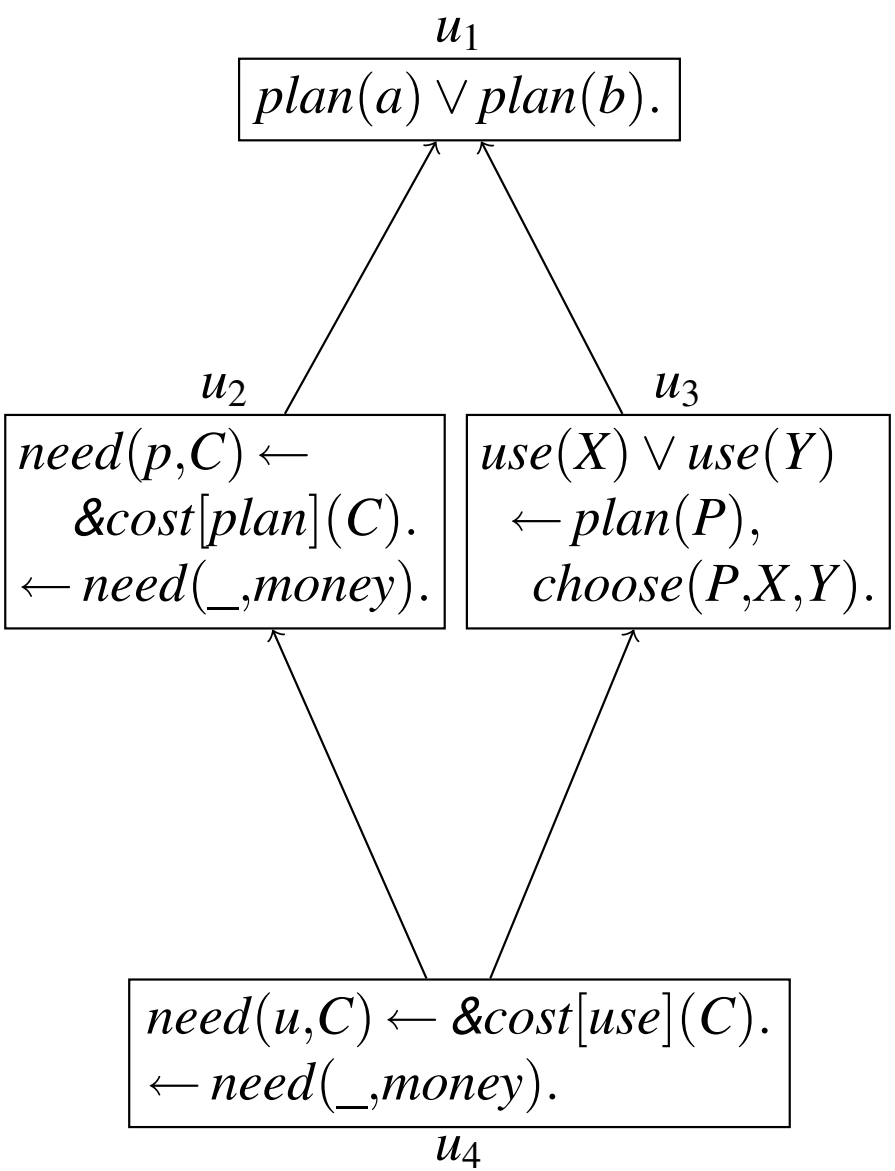


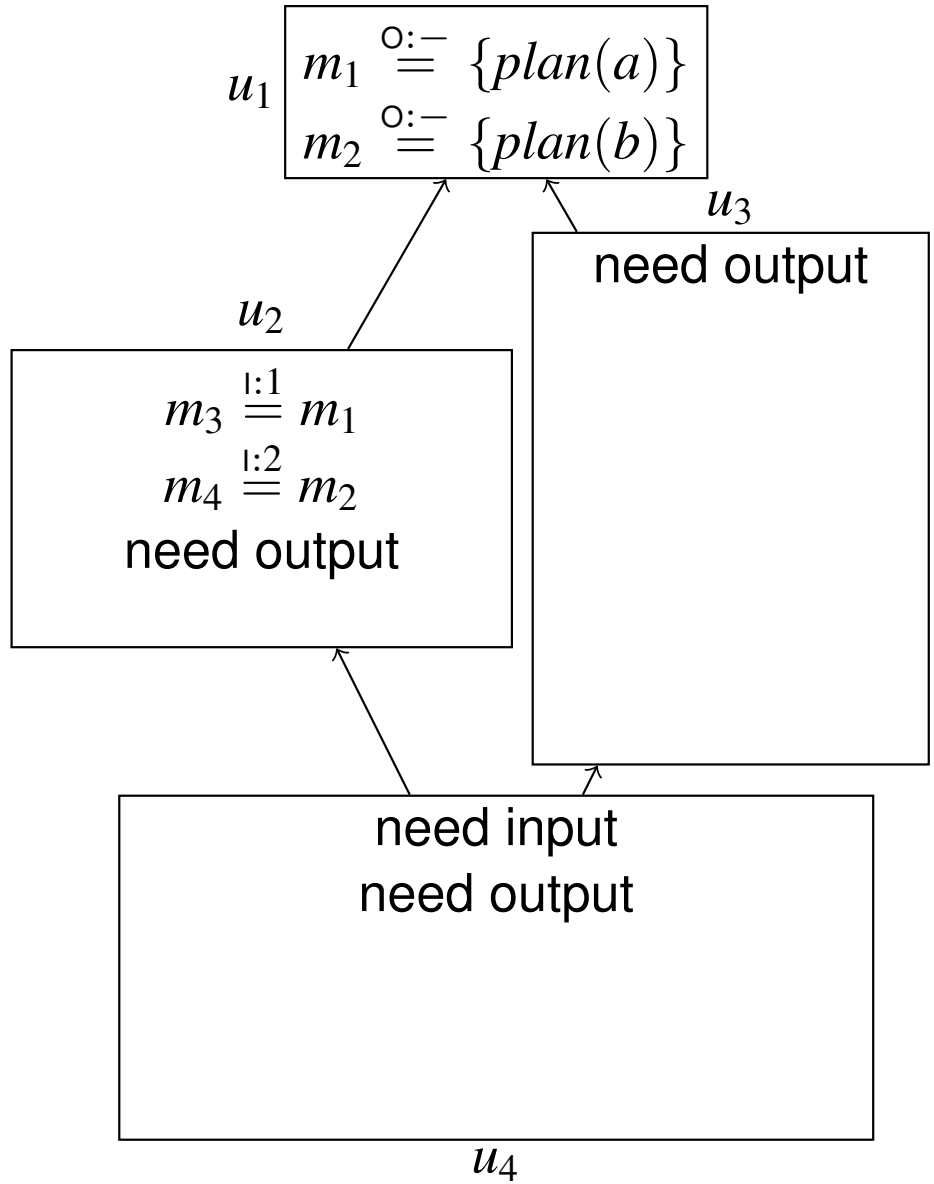
New Strategy Model Building



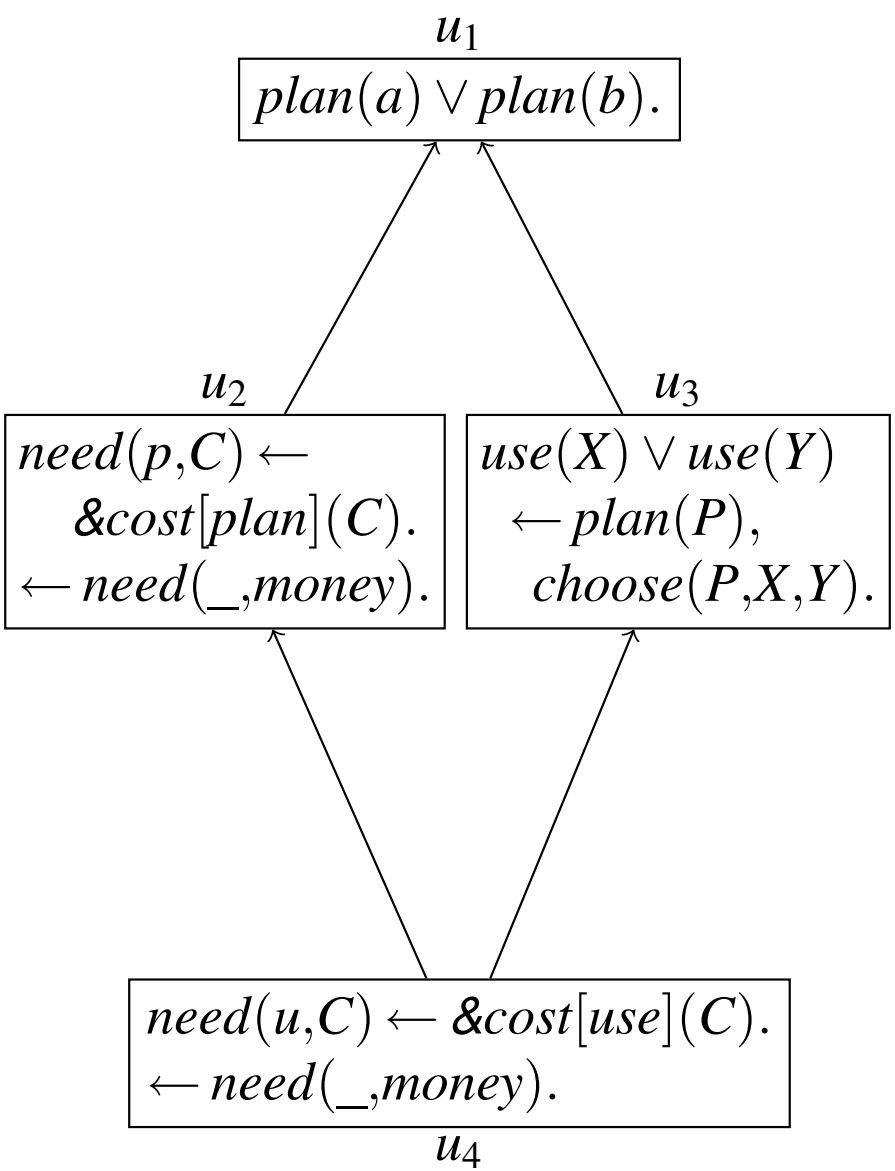


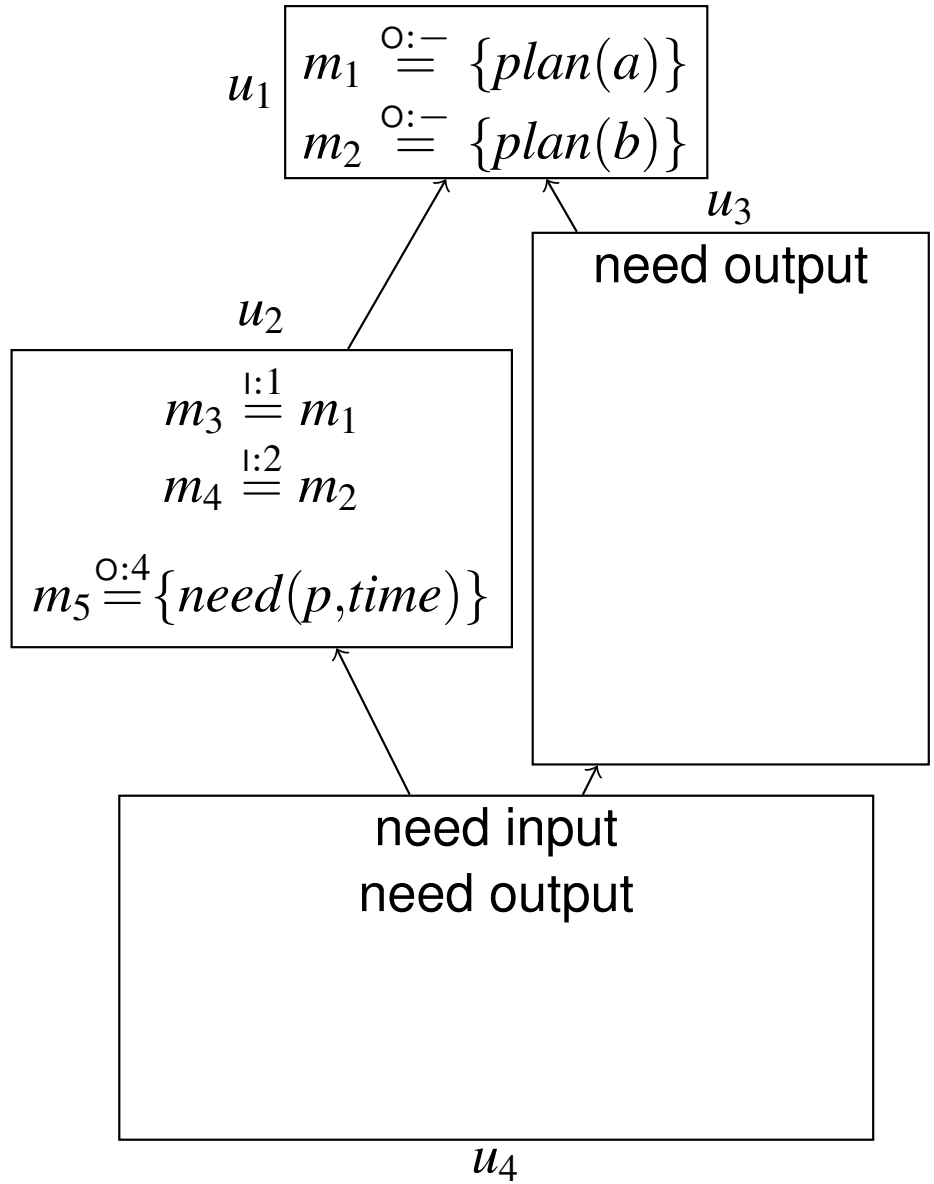
New Strategy Model Building



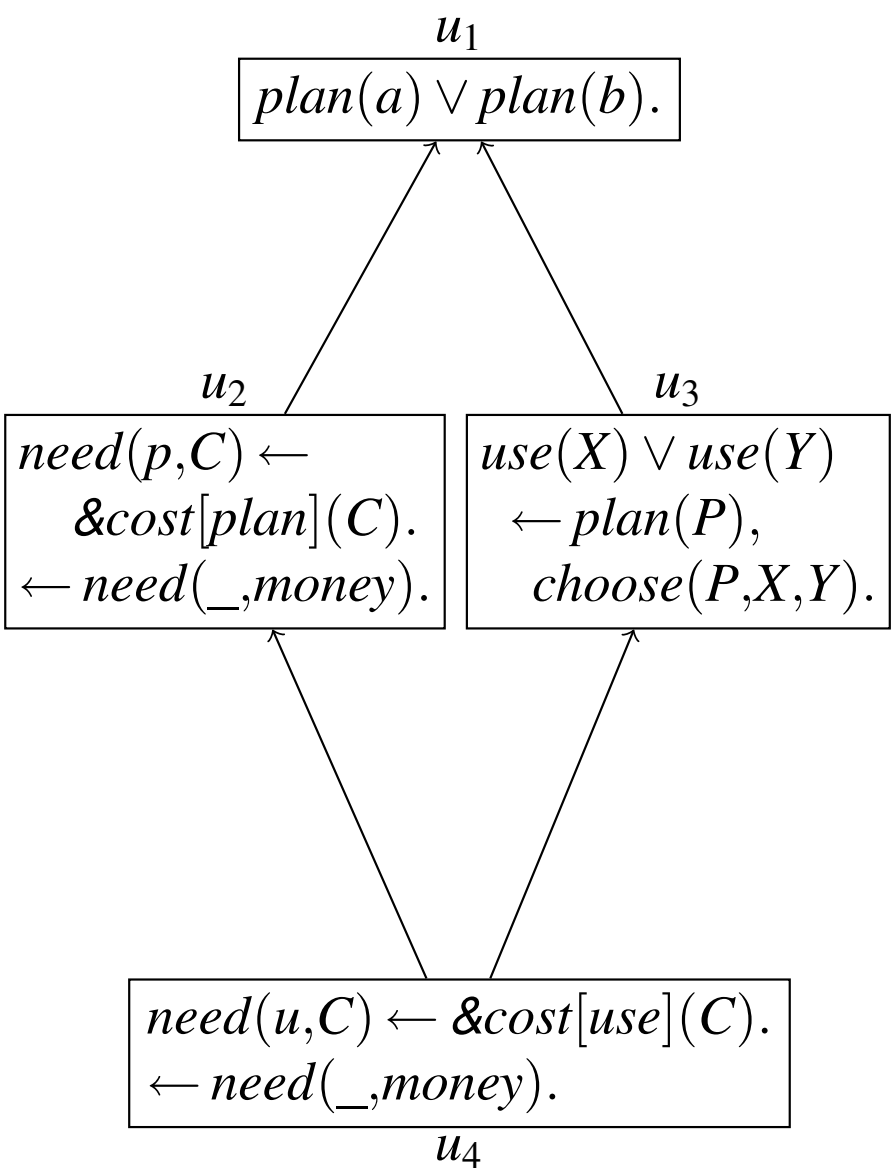


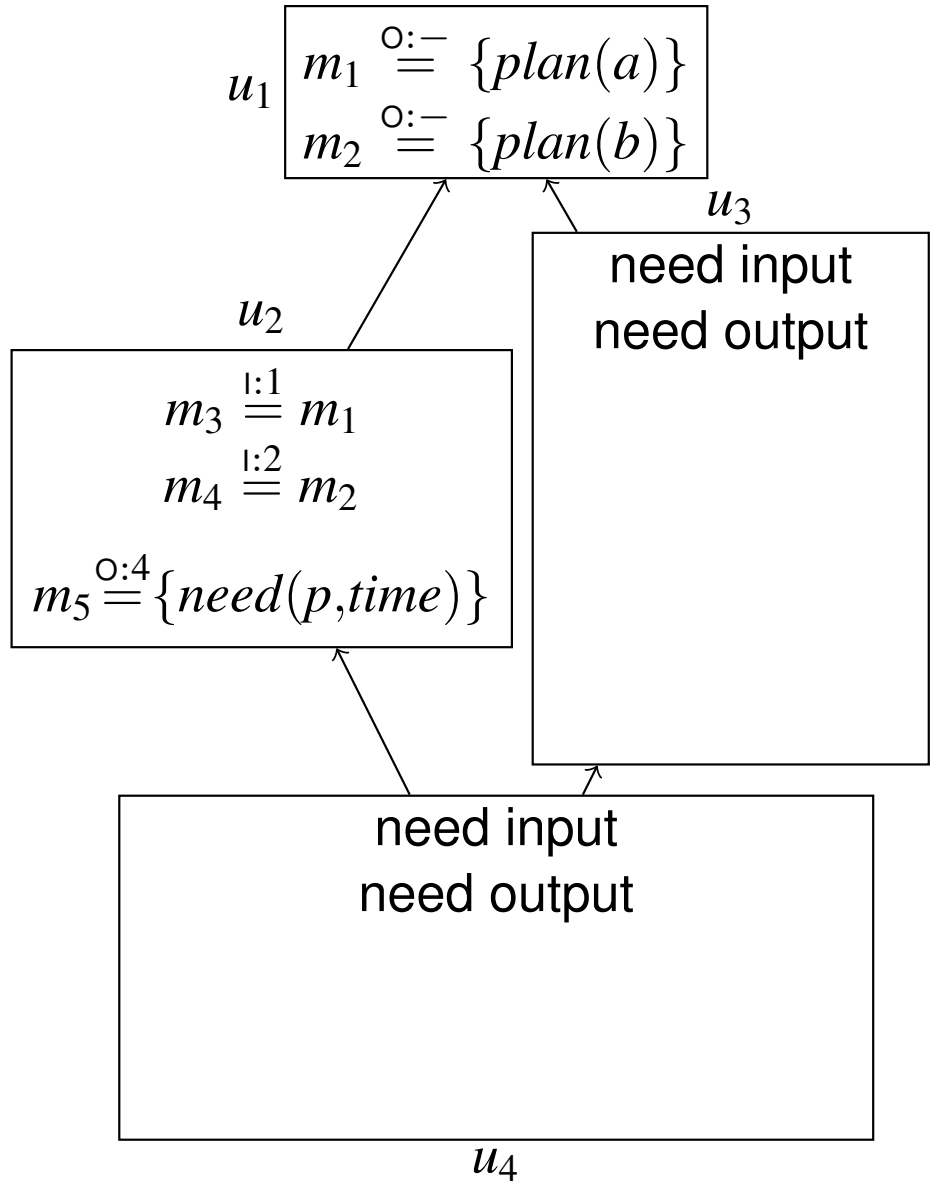
New Strategy Model Building



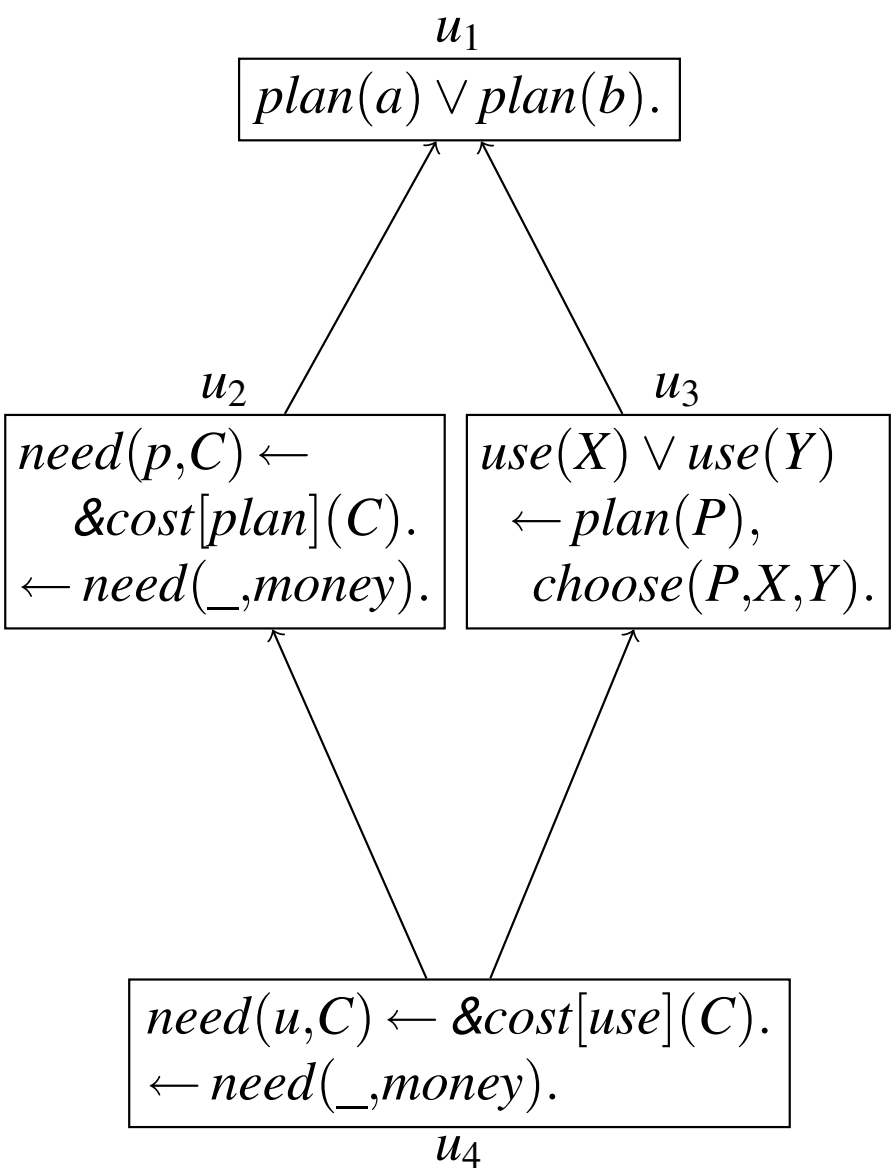


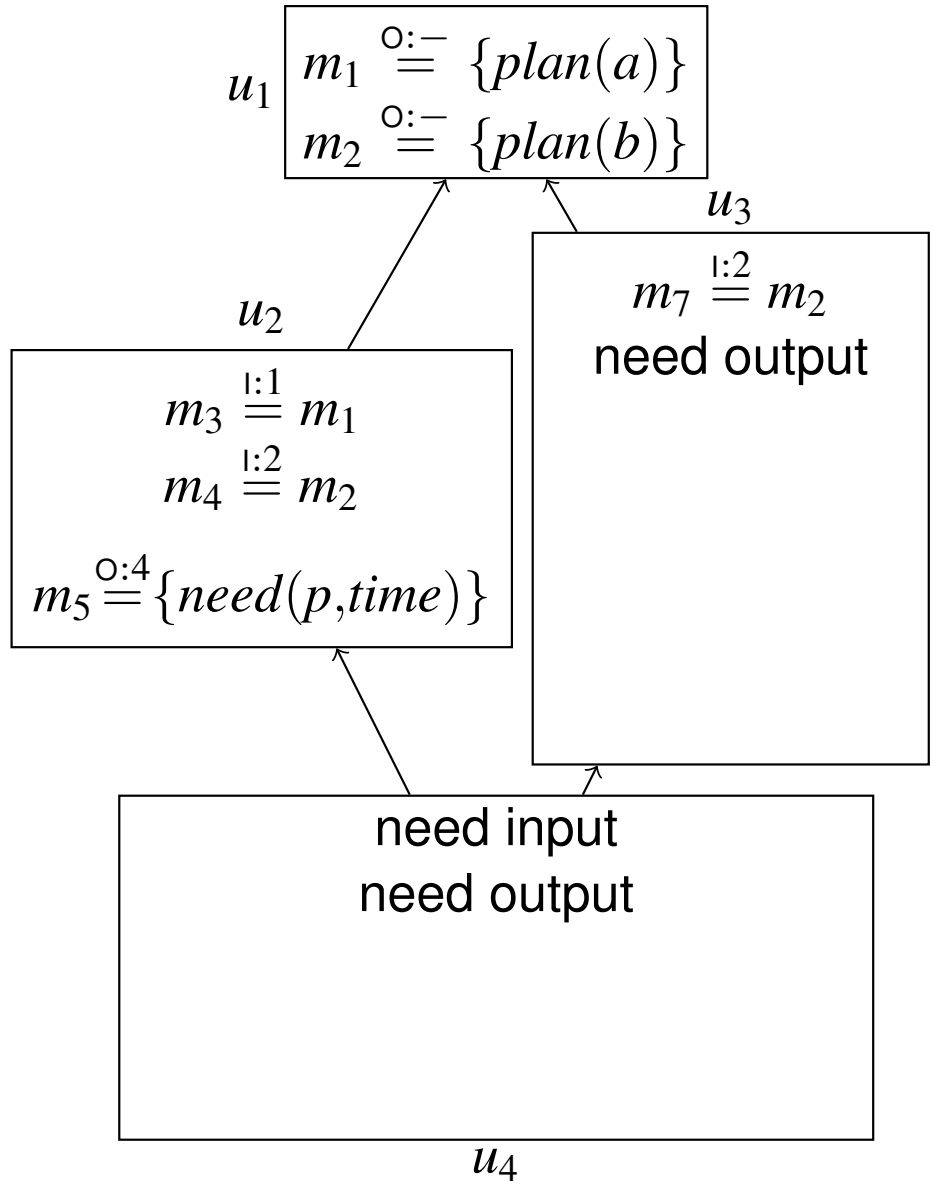
New Strategy Model Building



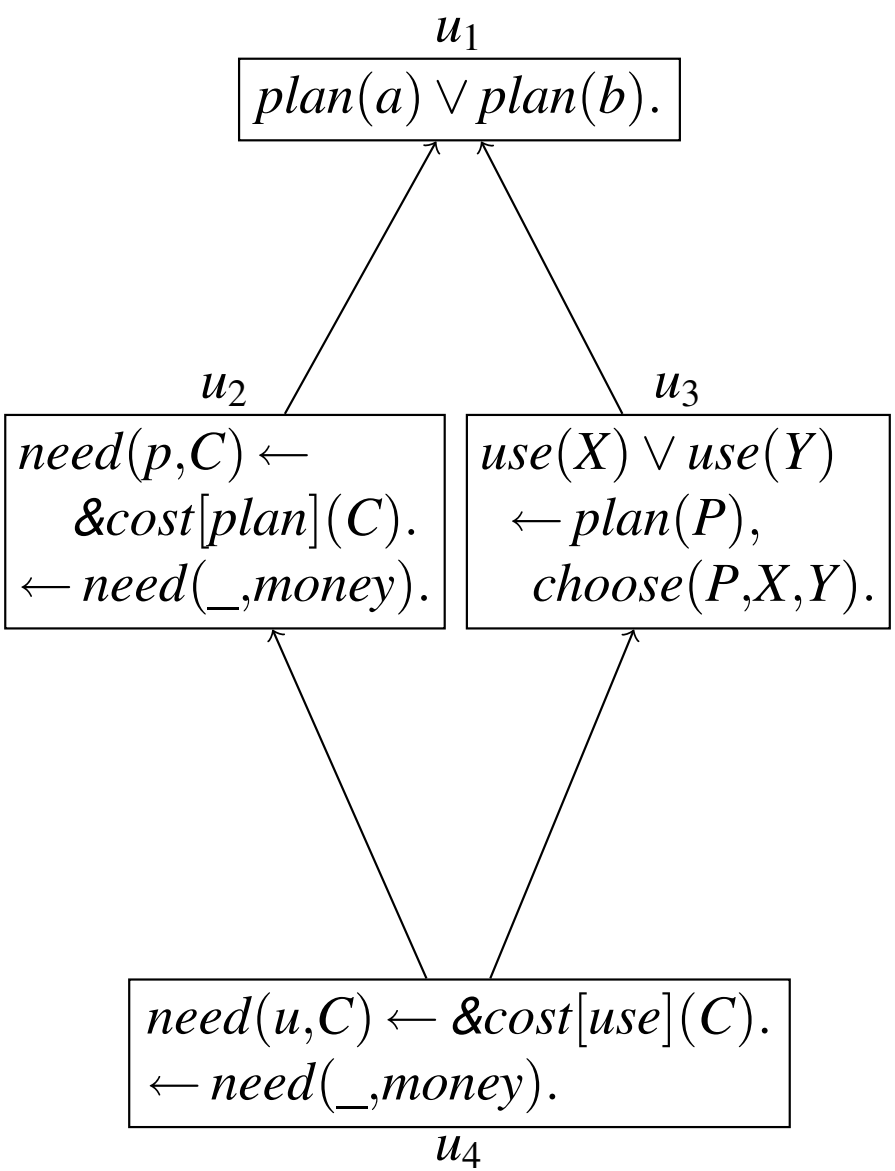


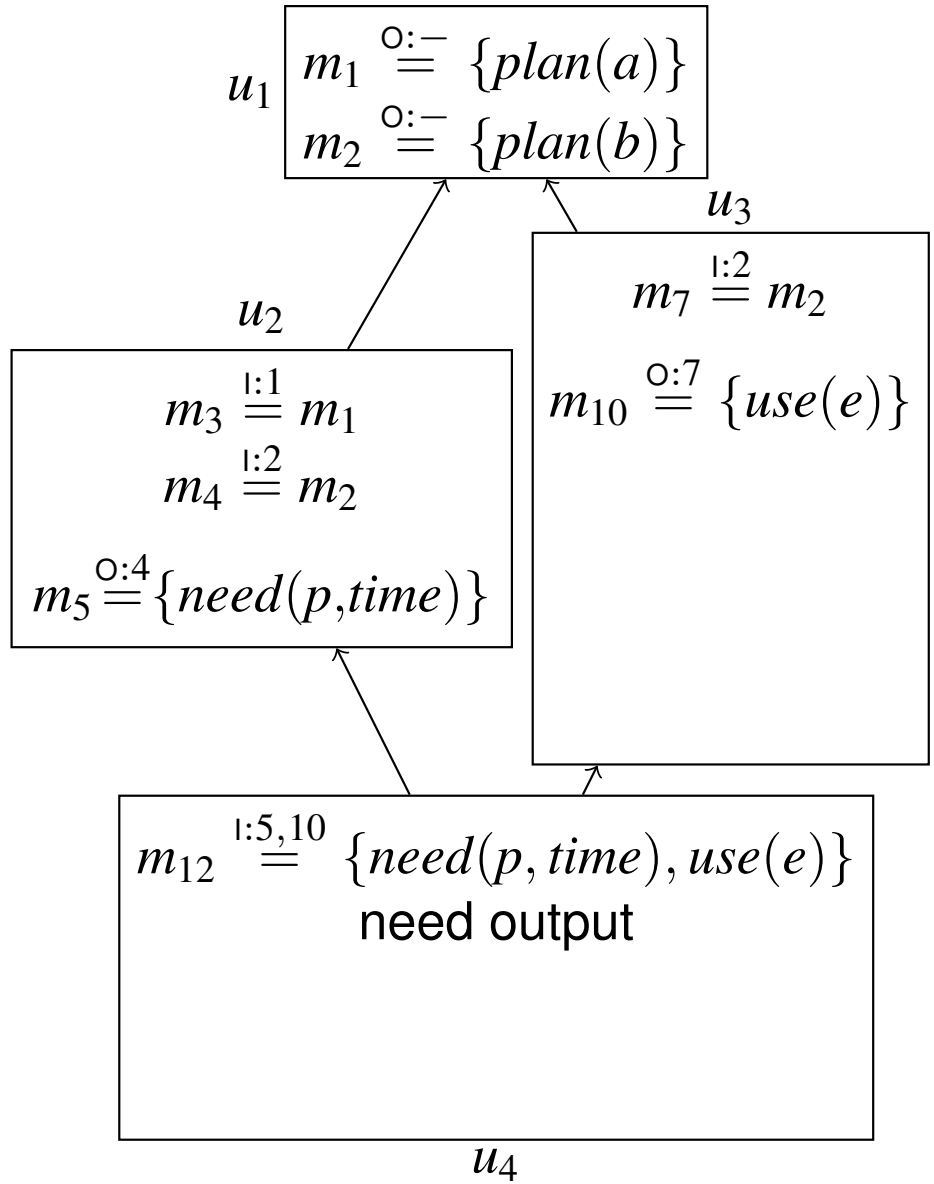
New Strategy Model Building



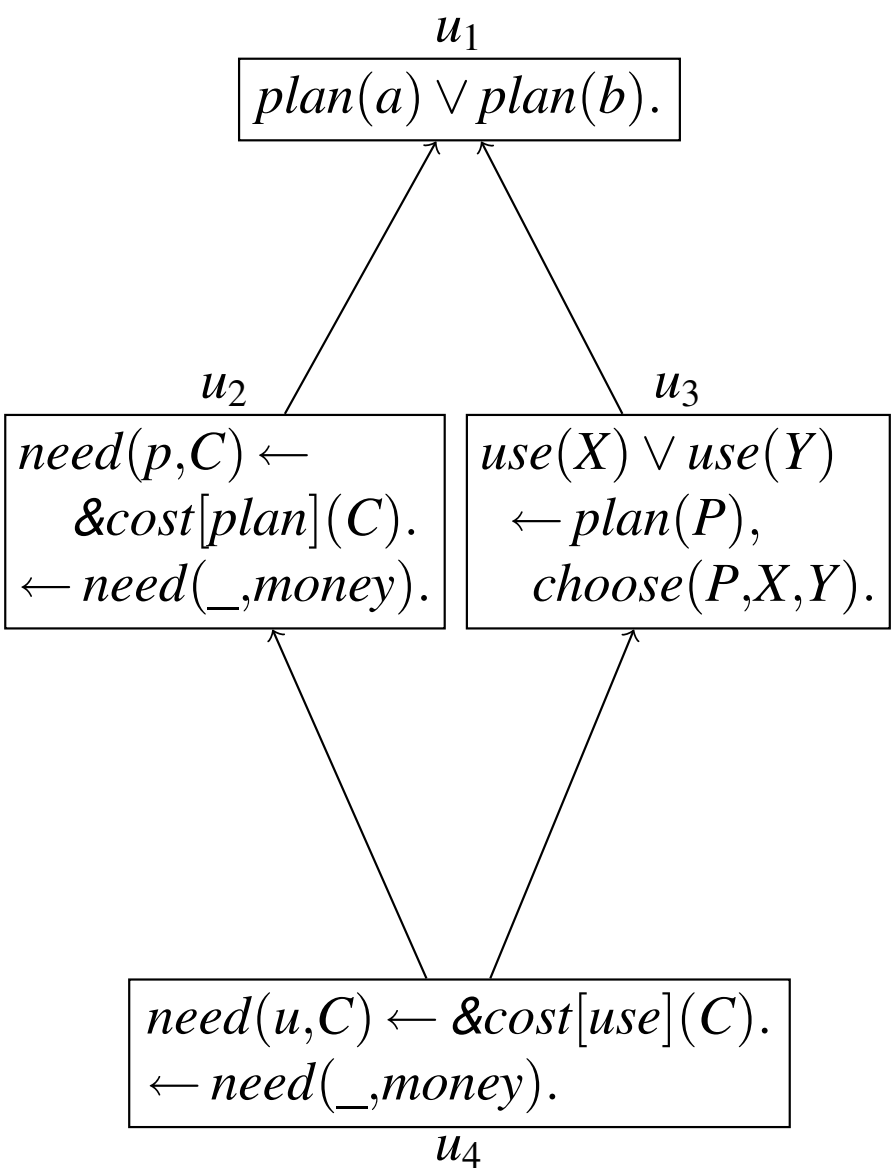


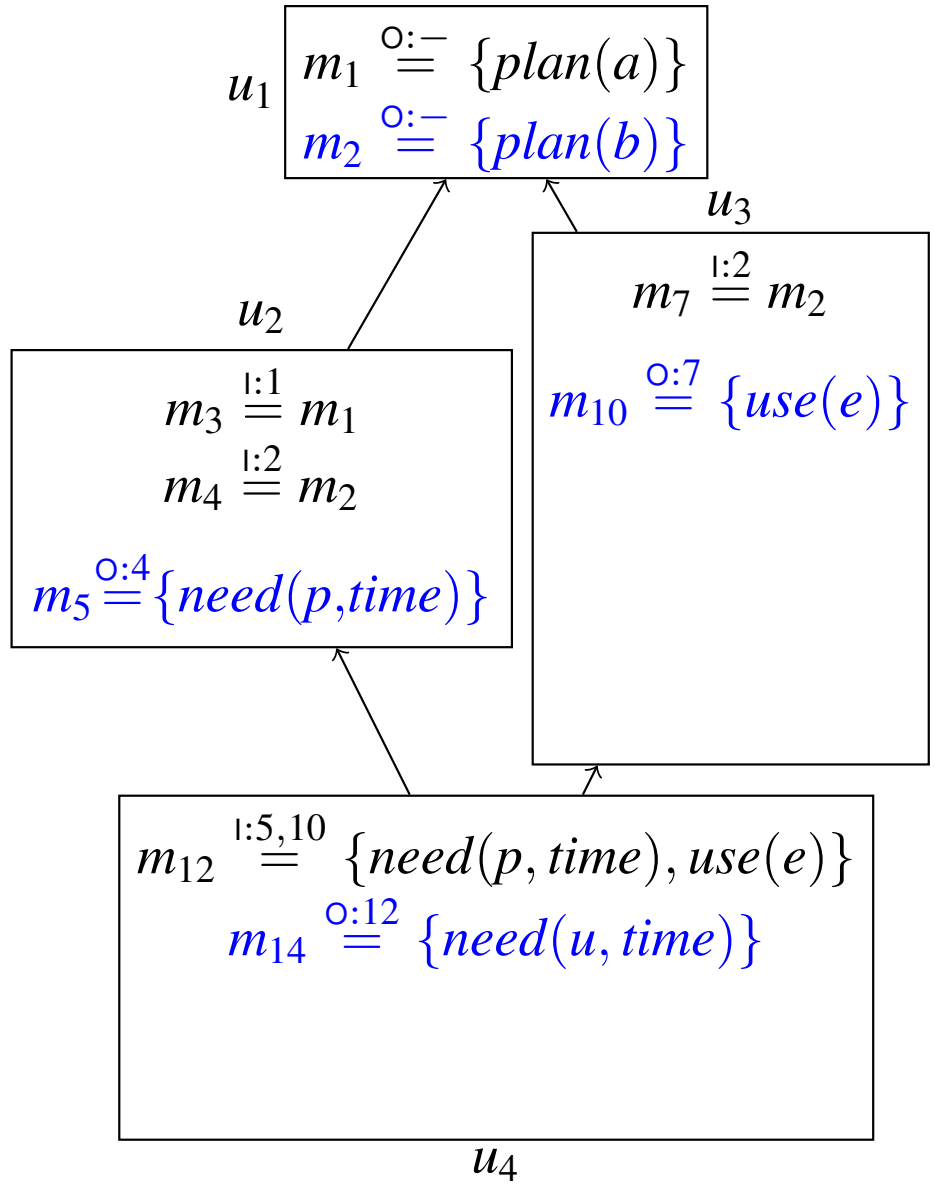
New Strategy Model Building



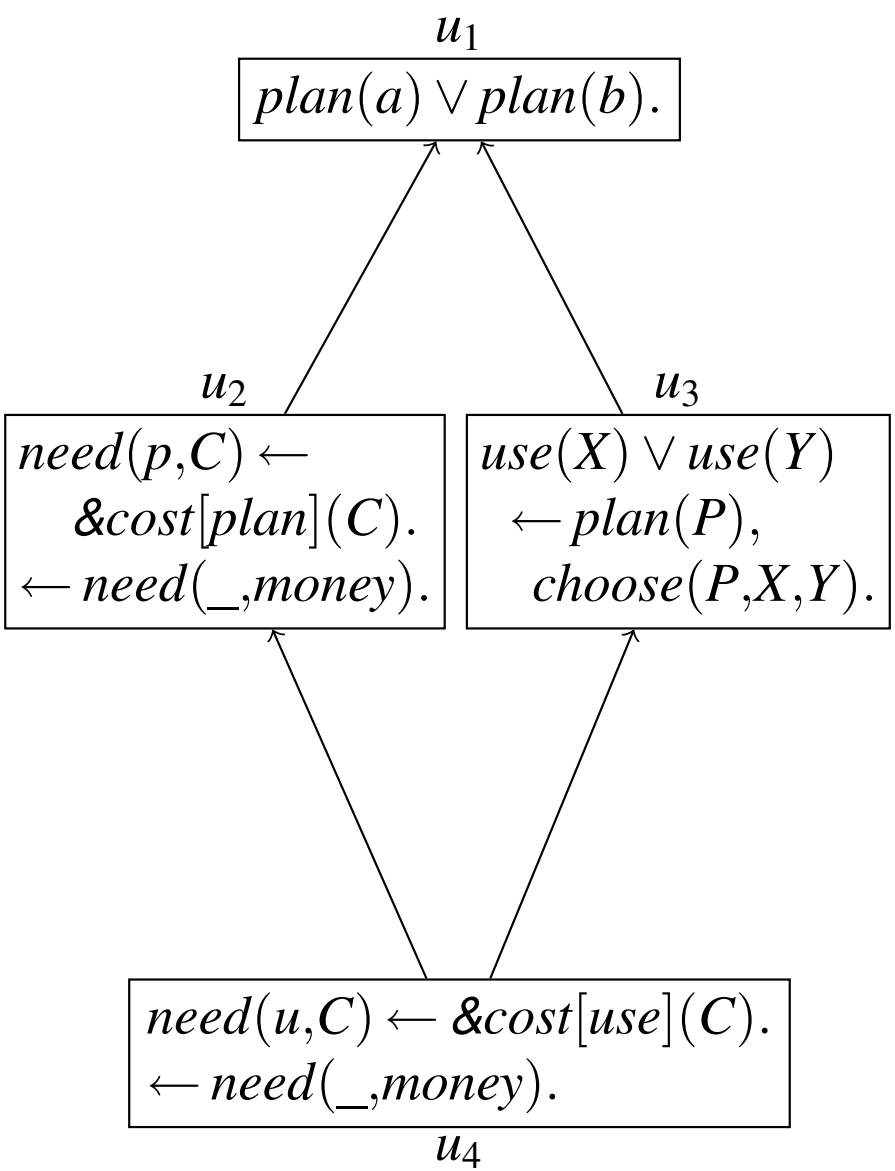


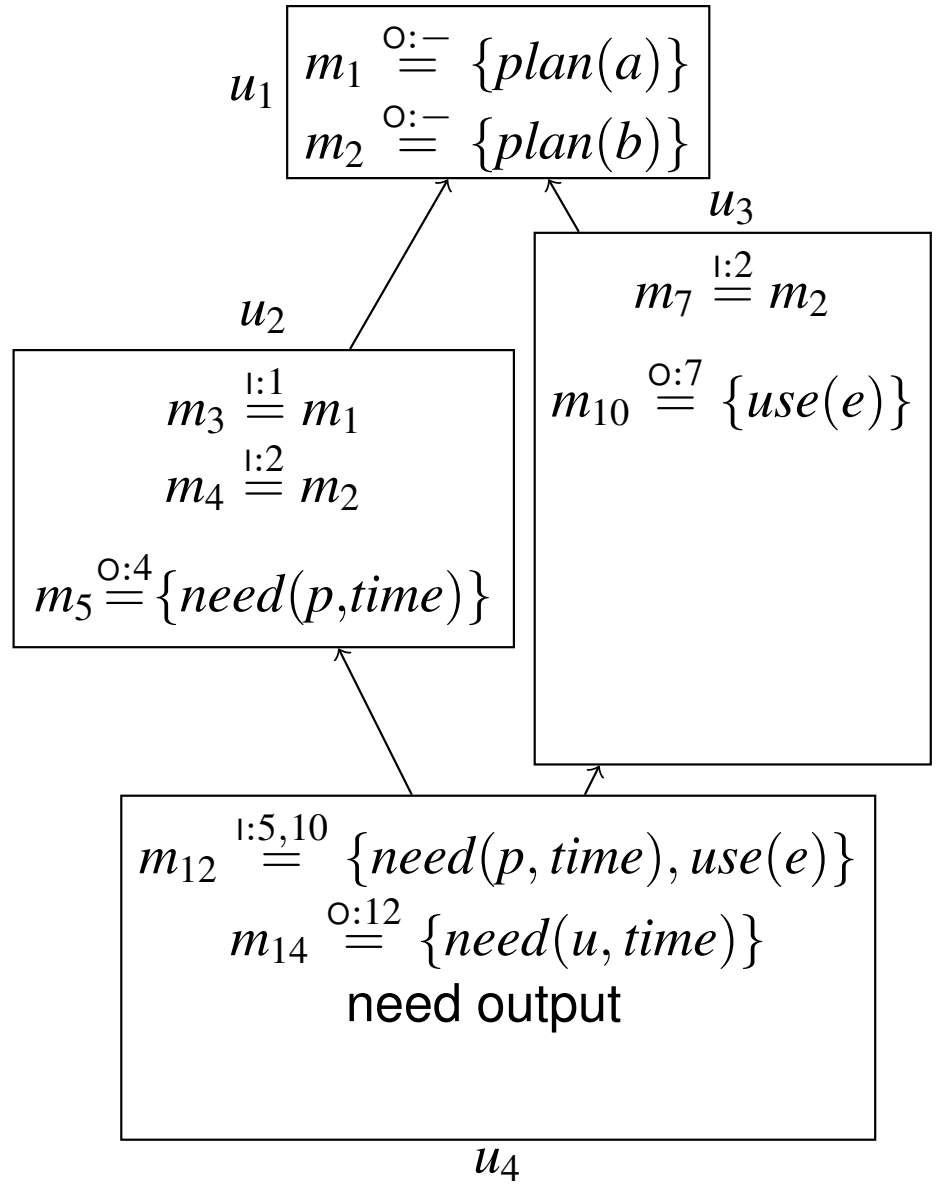
New Strategy Model Building



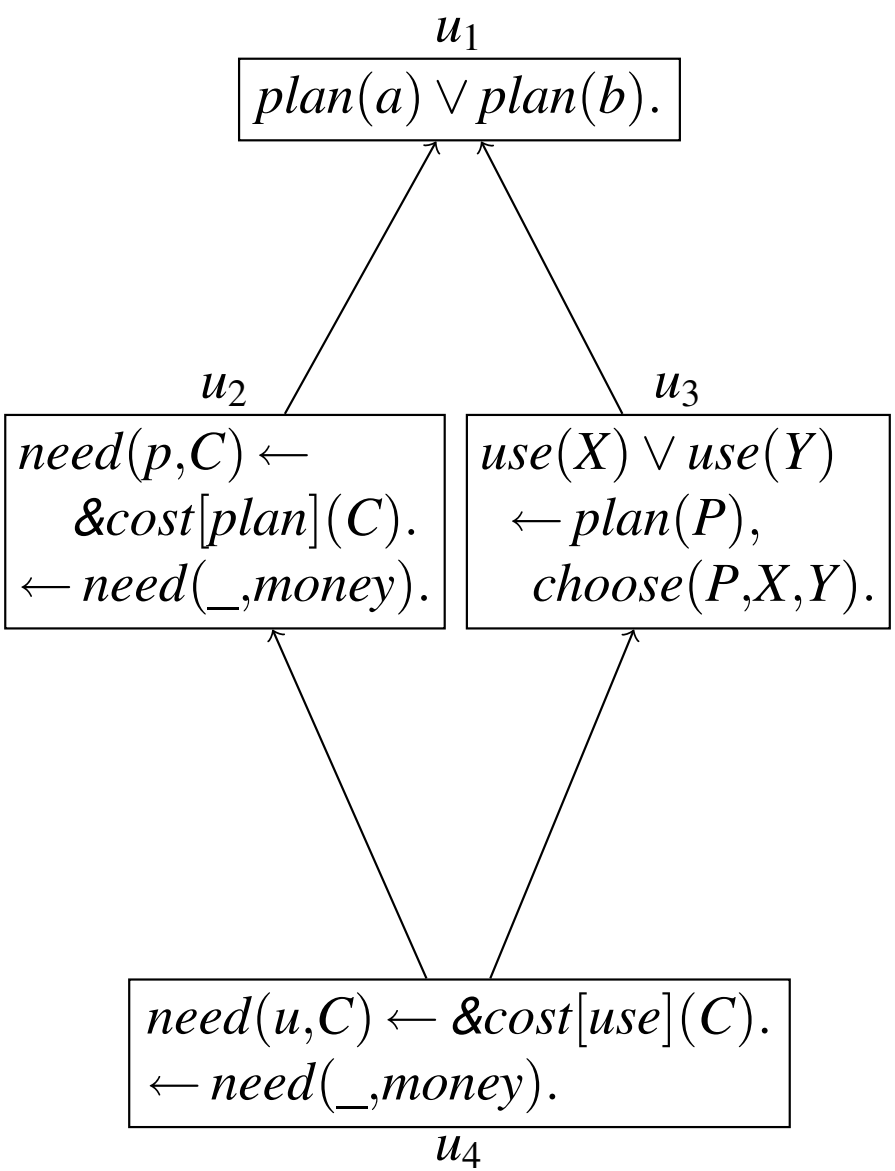


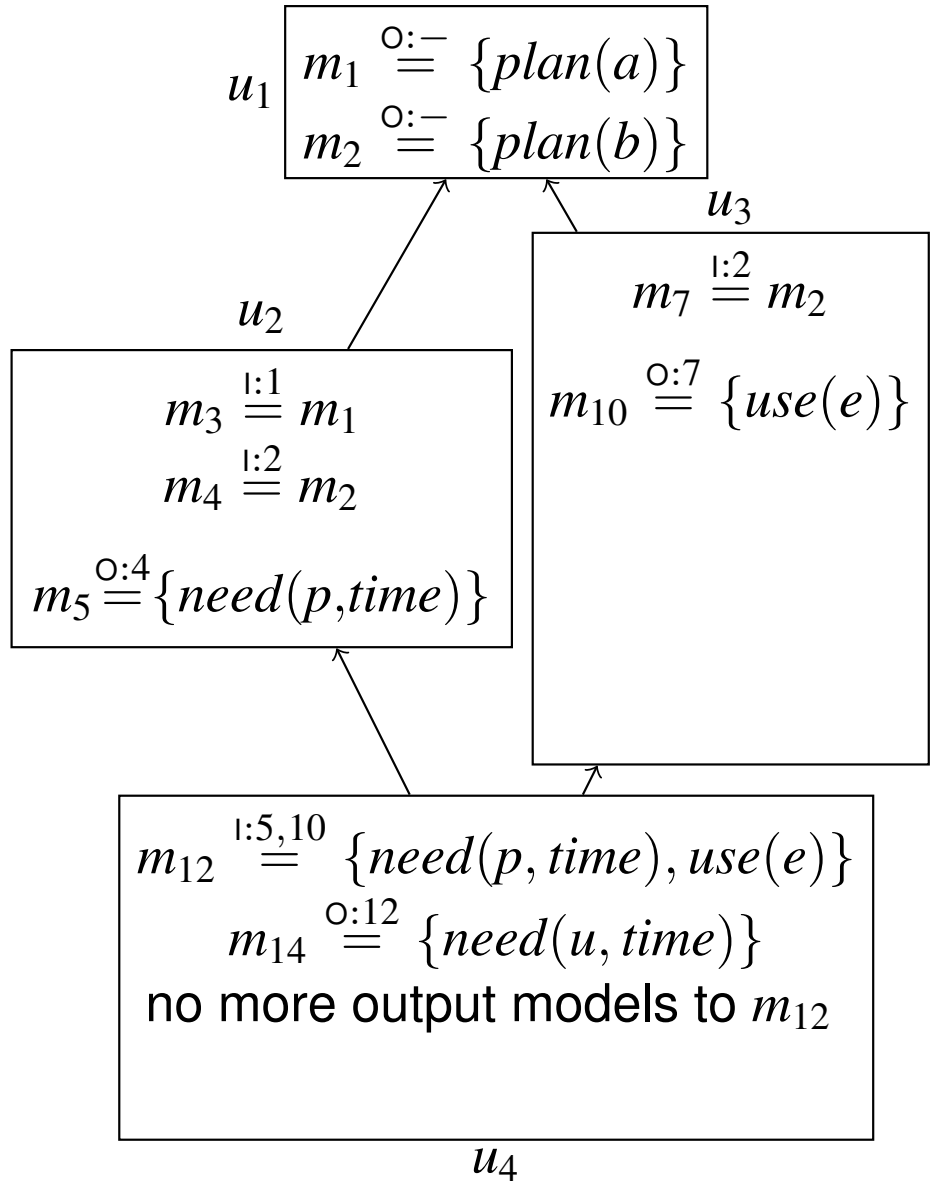
New Strategy Model Building



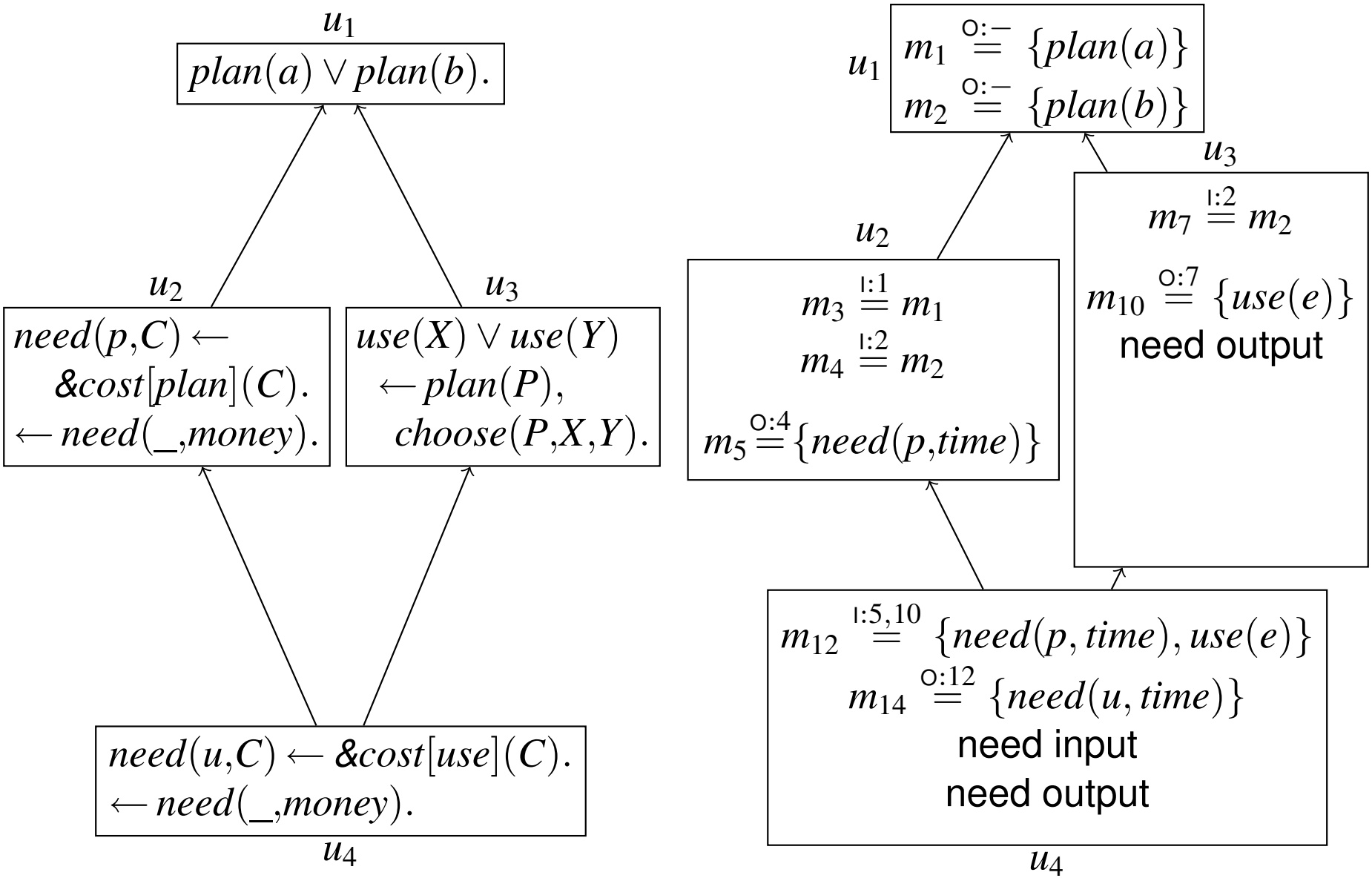


New Strategy Model Building

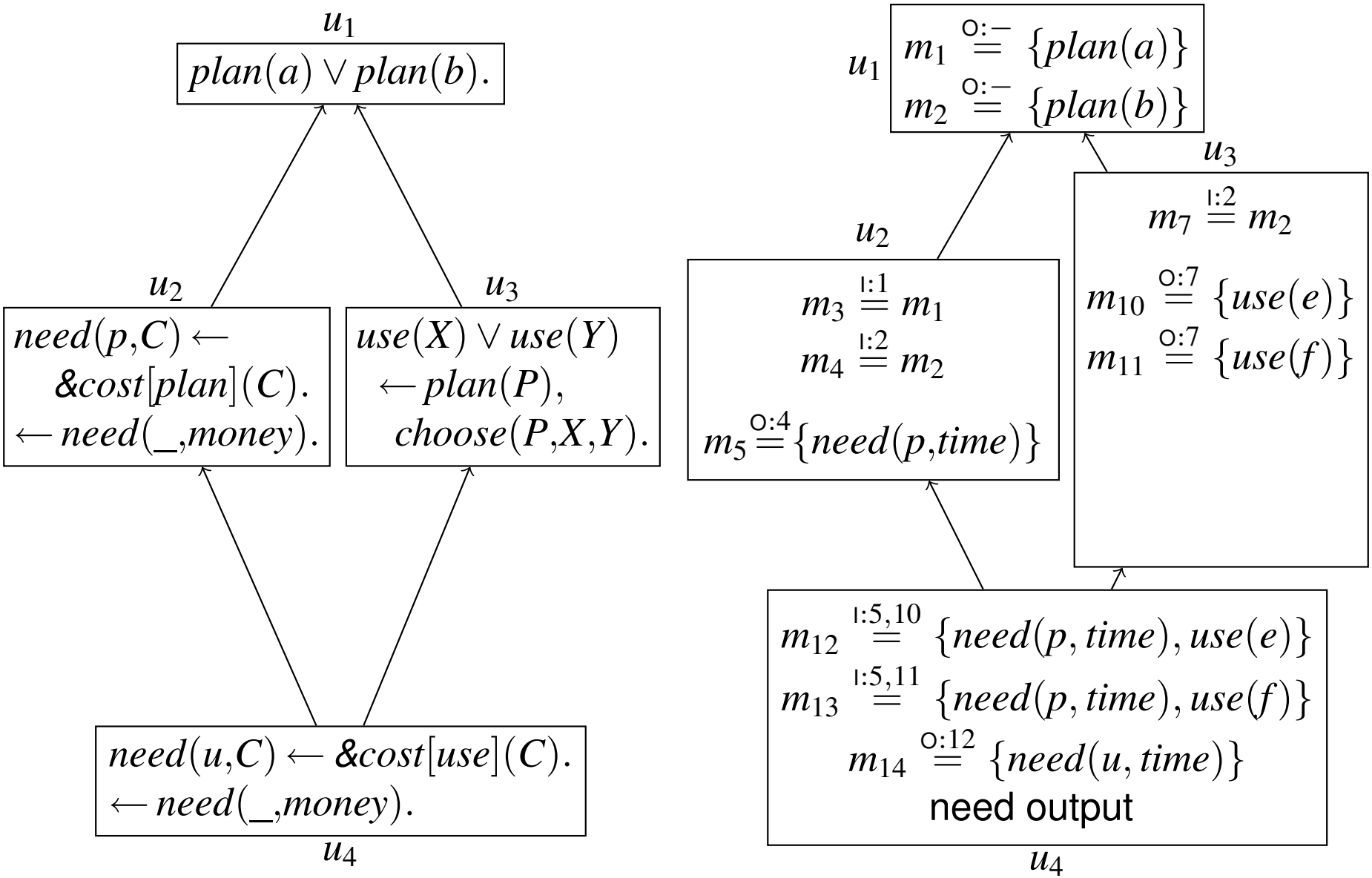




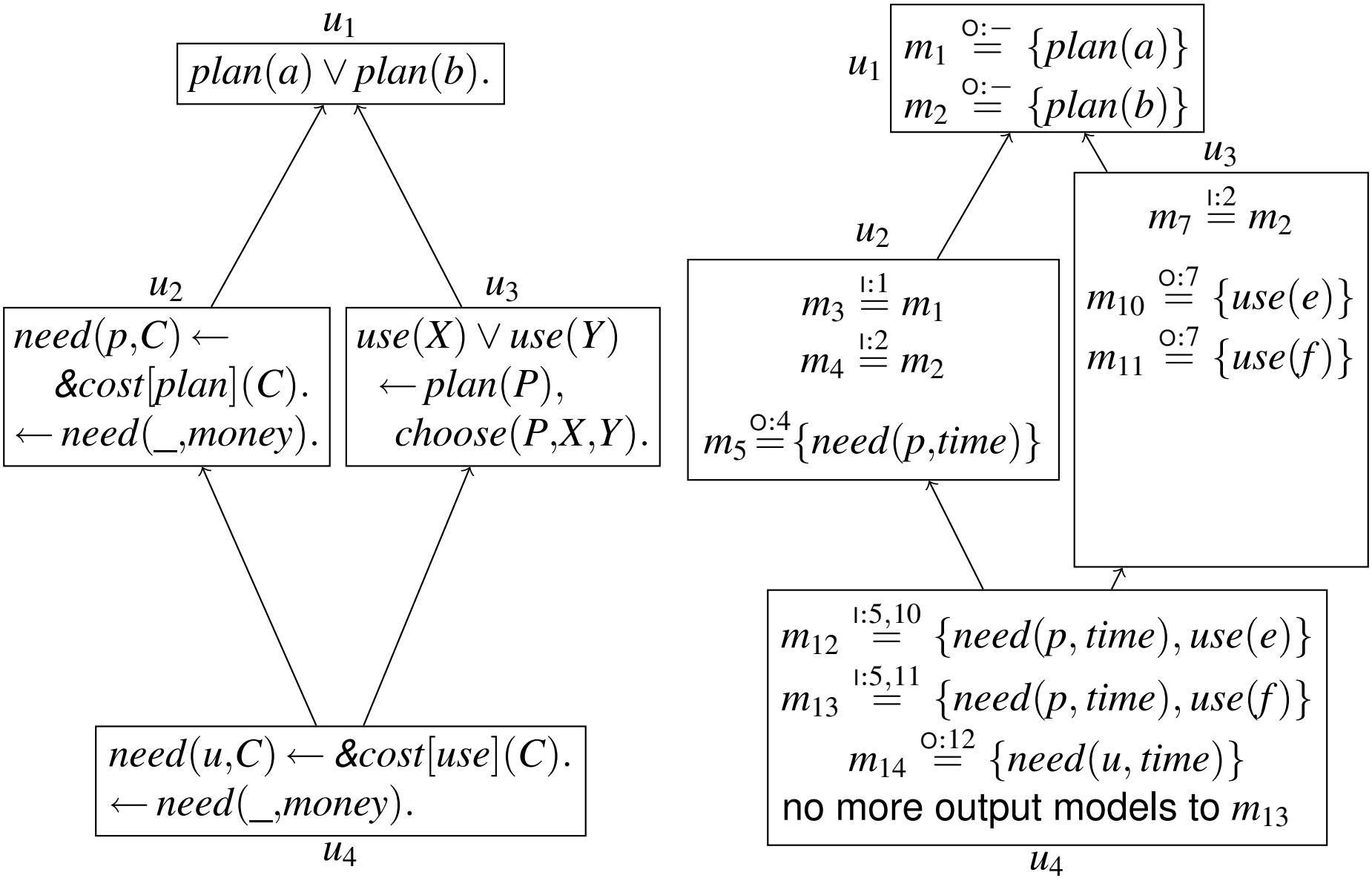
New Strategy Model Building



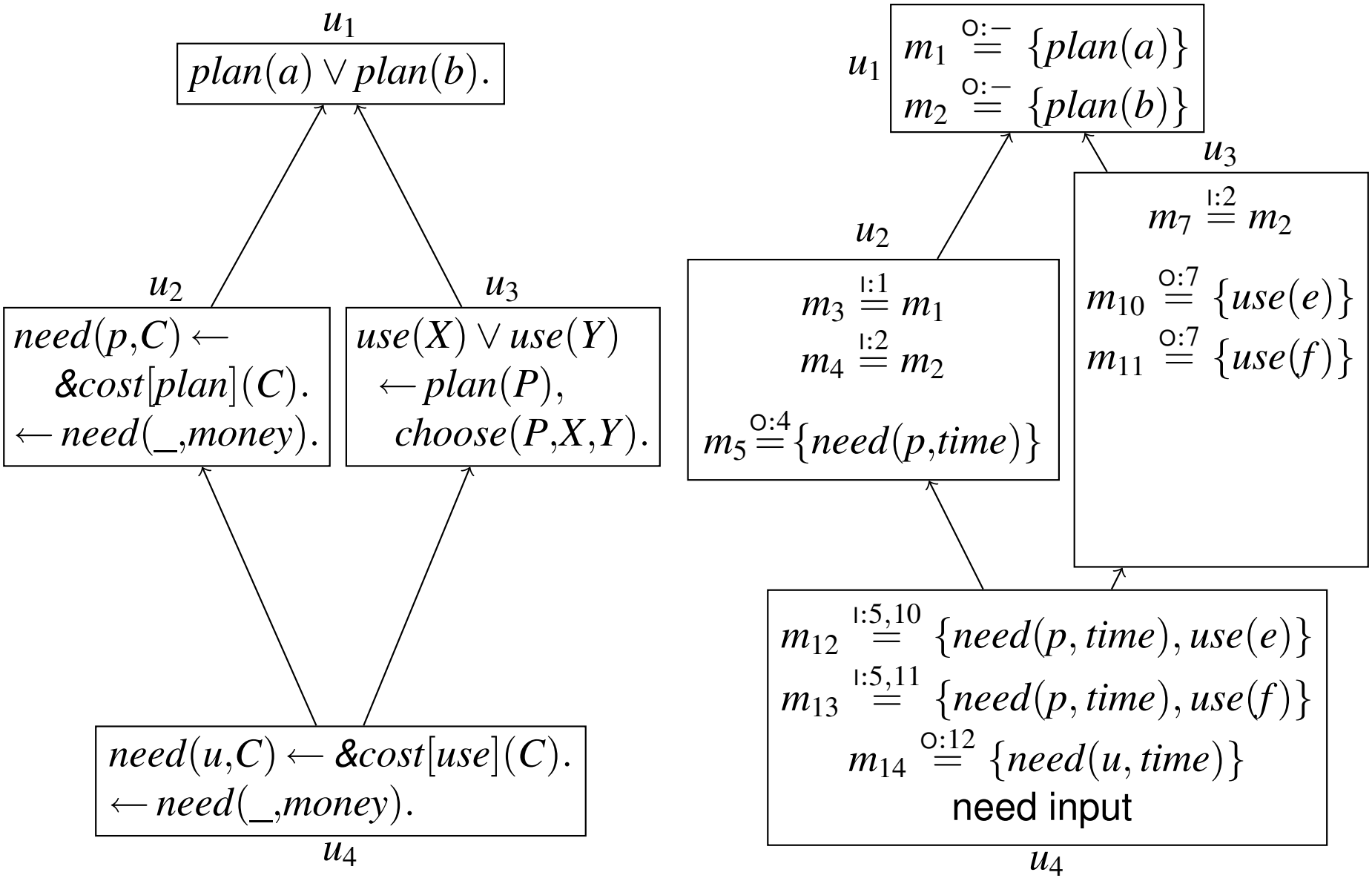
New Strategy Model Building



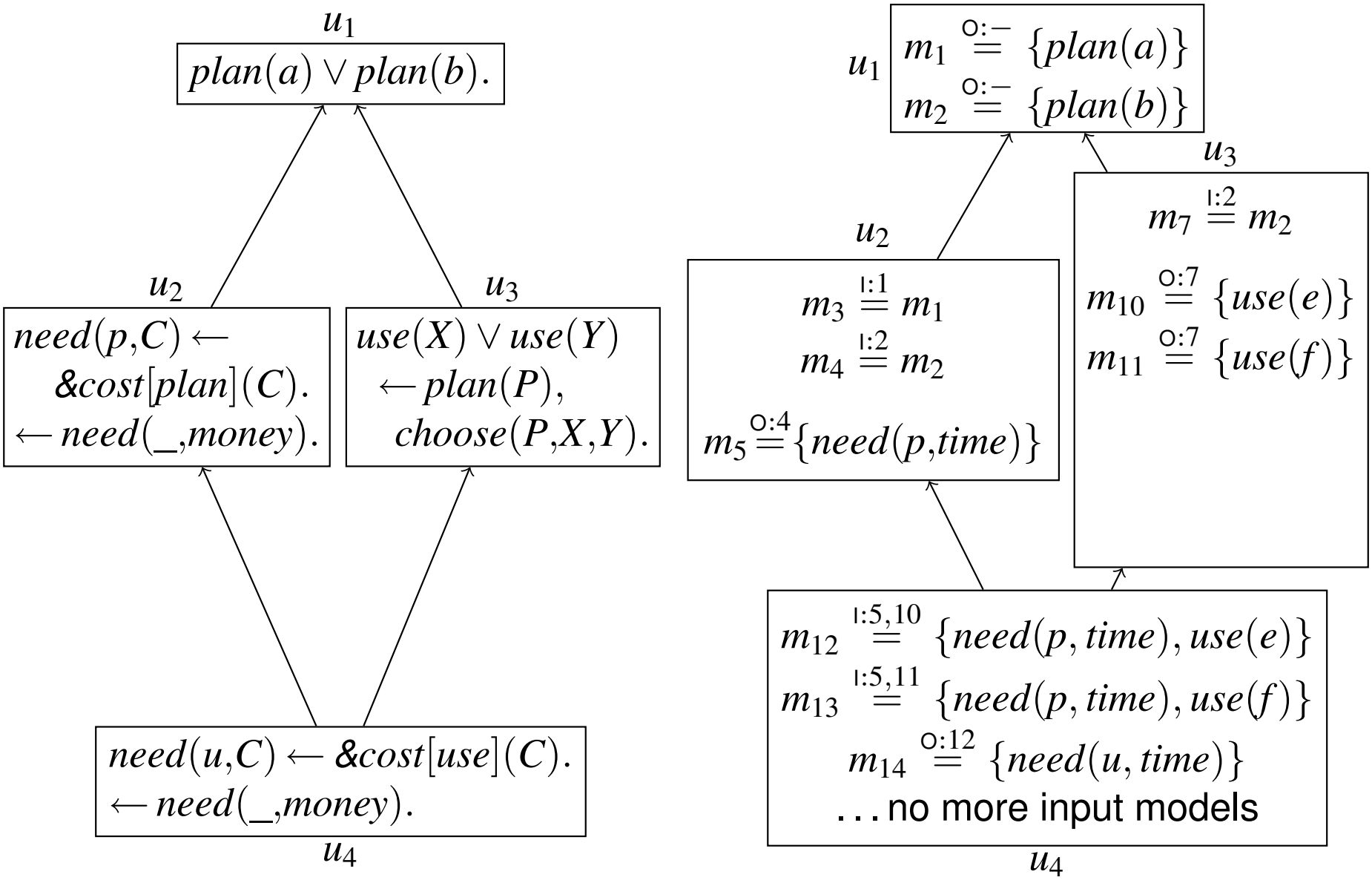
New Strategy Model Building



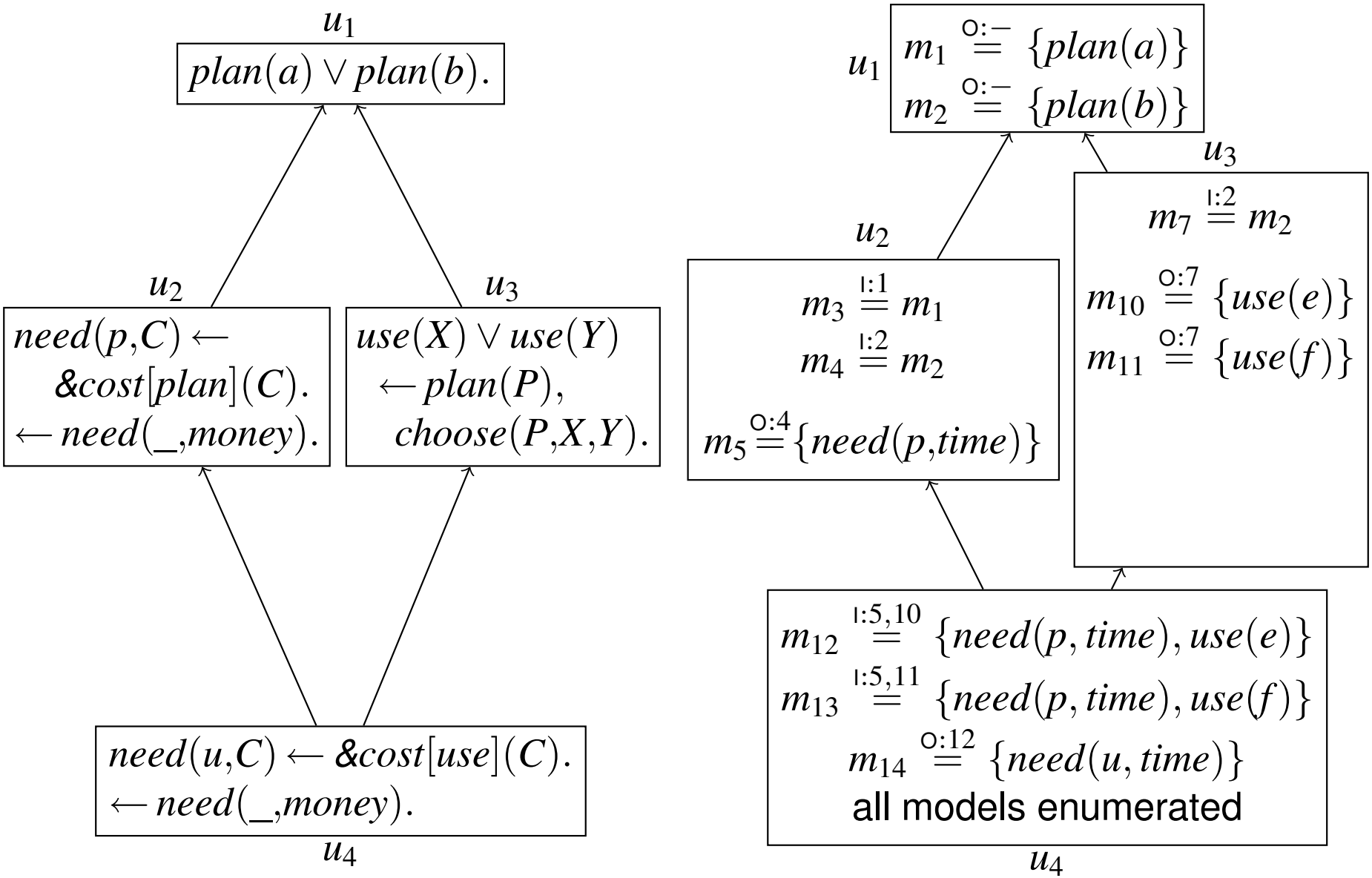
New Strategy Model Building



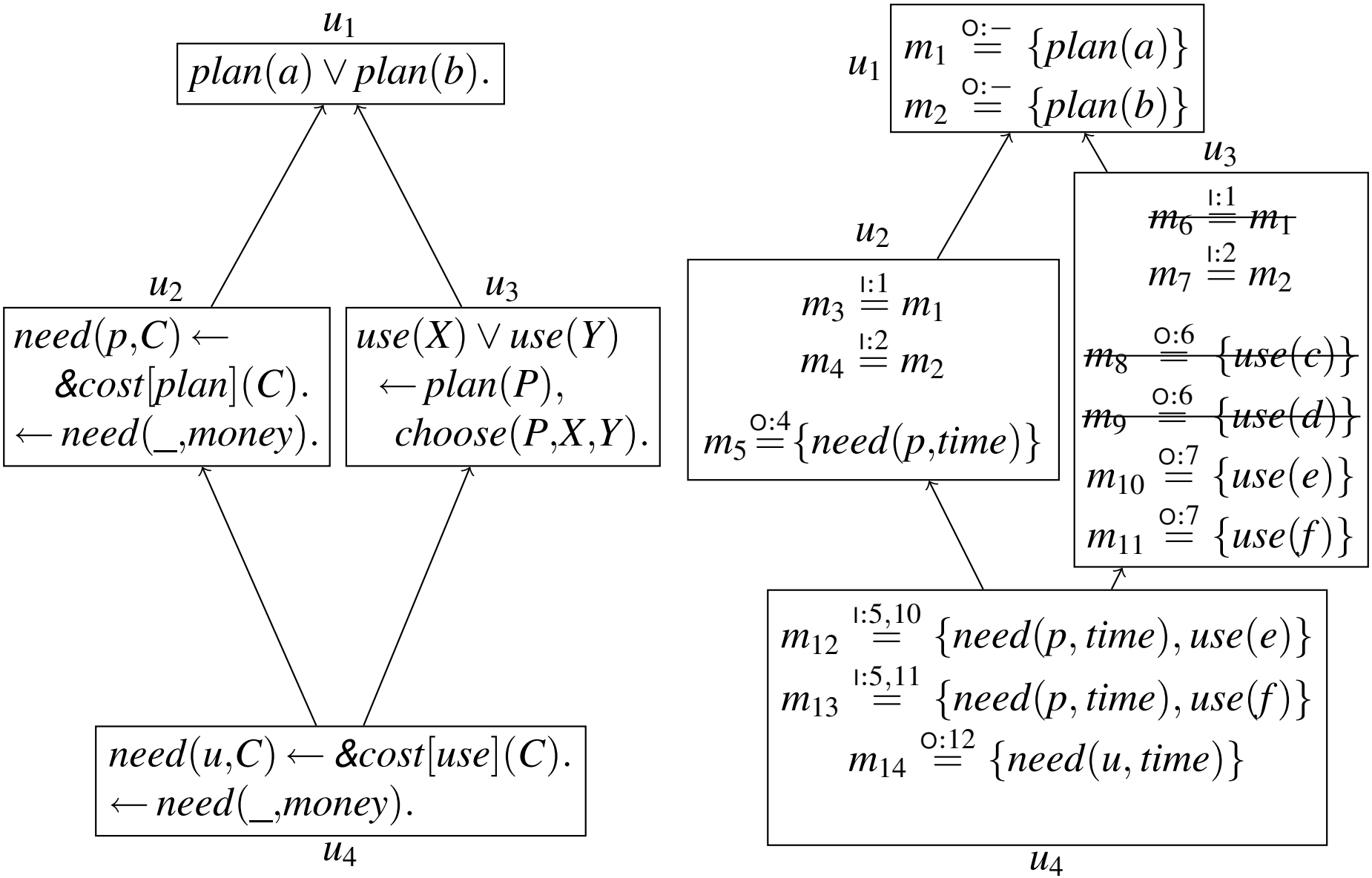
New Strategy Model Building



New Strategy Model Building



New Strategy Model Building



Evaluation Algorithm Improvements

Acyclic Directed Evaluation Graph

- $\Rightarrow$ independent program fragments evaluated independently
- $\Rightarrow$ fewer redundant evaluations
- $\Rightarrow$ parallel evaluation possible

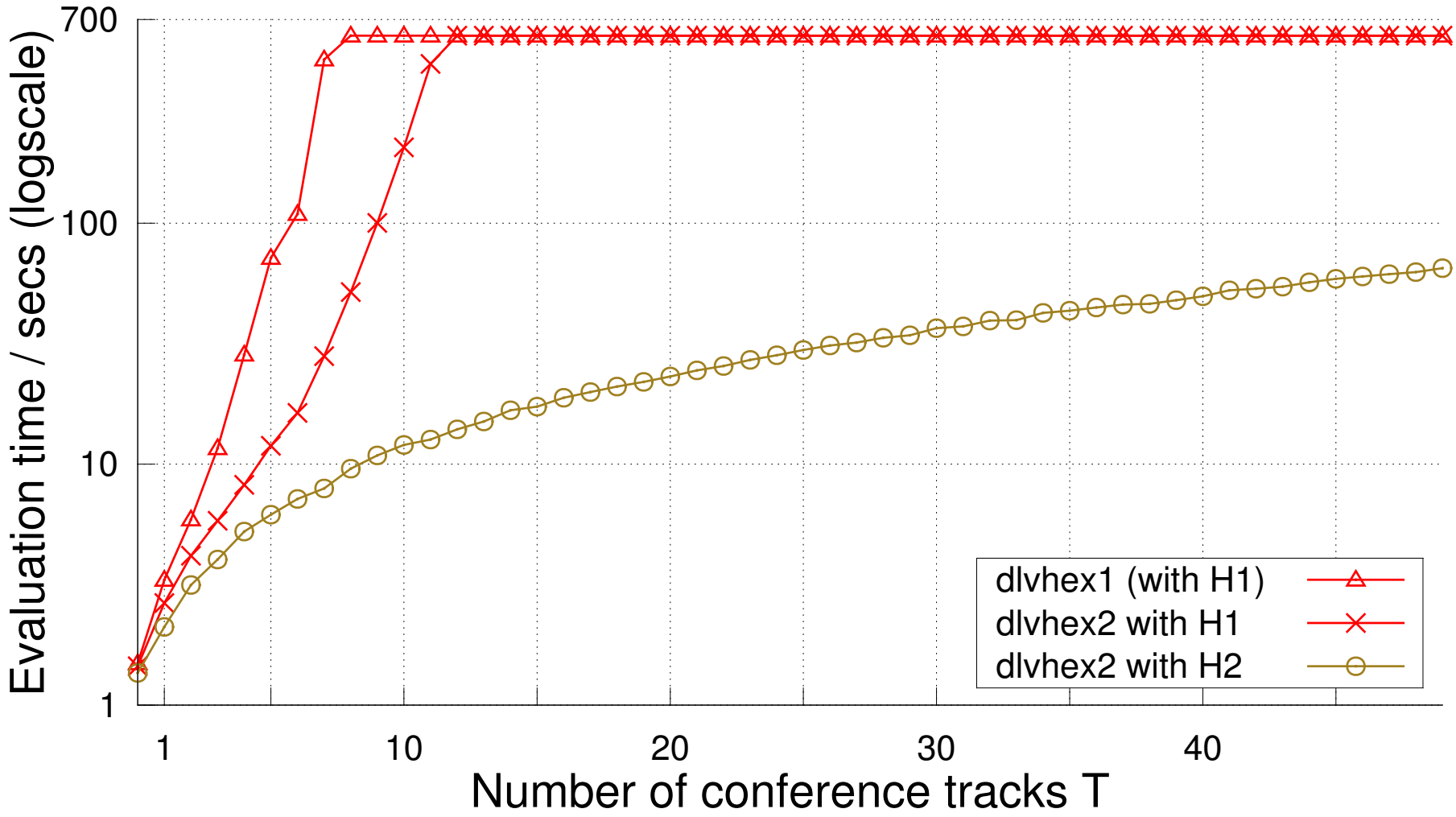
Constraints can be shared between units

- $\Rightarrow$ early model elimination

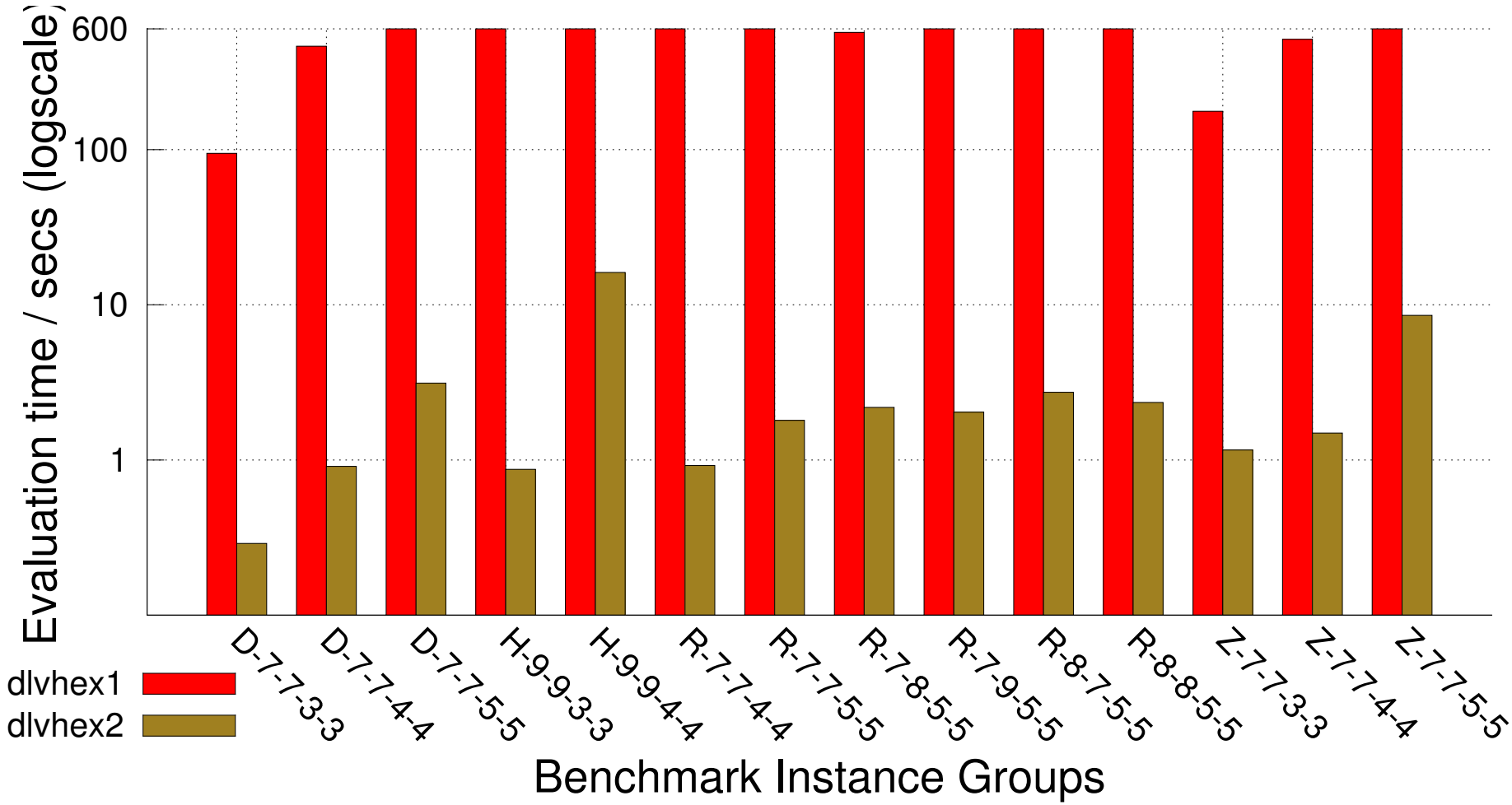
Implementation Improvements

- Model generation as streaming mode
  - $\Rightarrow$ time-efficient first model computation
  - $\Rightarrow$ space-efficient model enumeration
    (output answer set $\longrightarrow$ delete part of graph)
  - $\Rightarrow$ 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

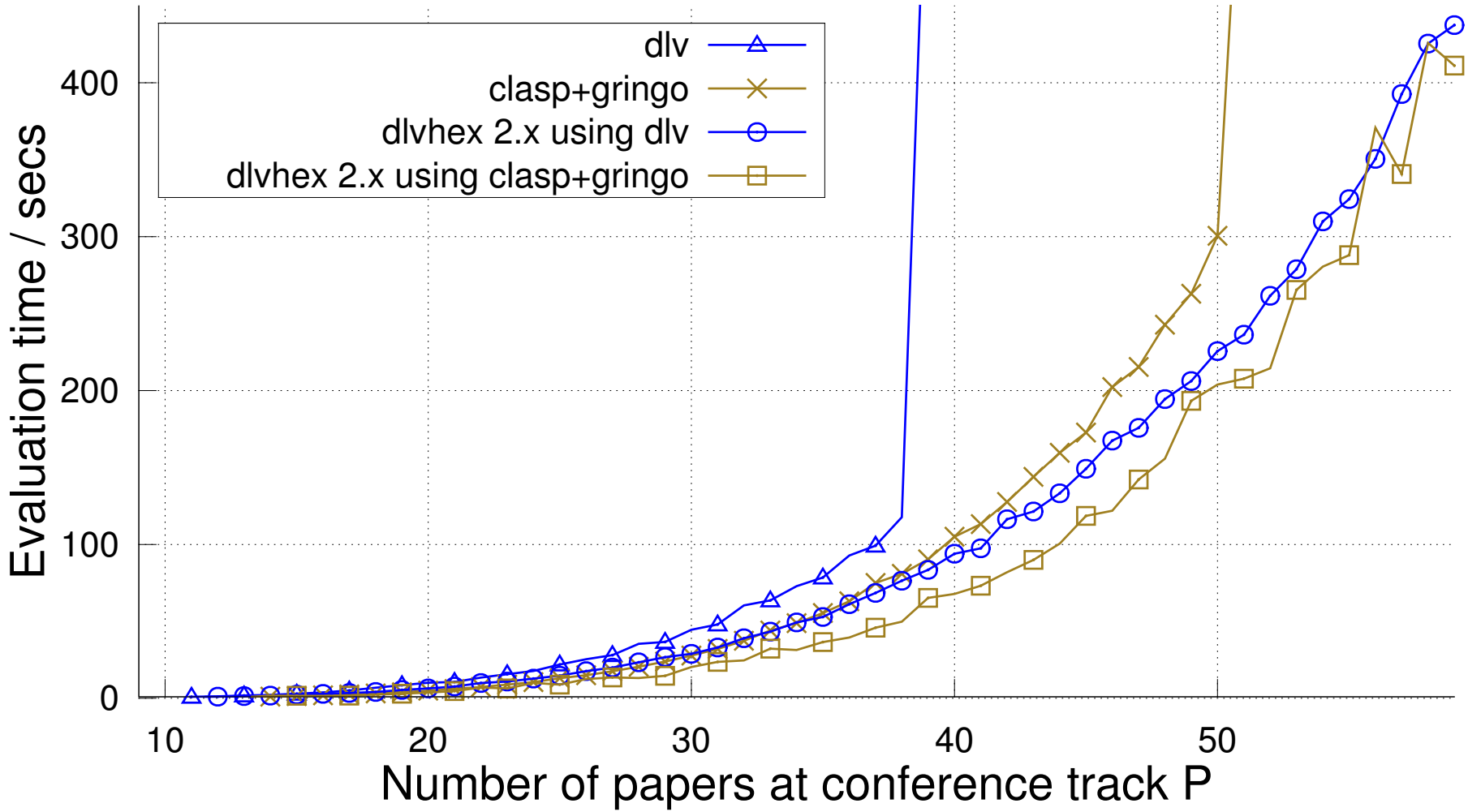
Benchmarks – Reviewer Selection HEX



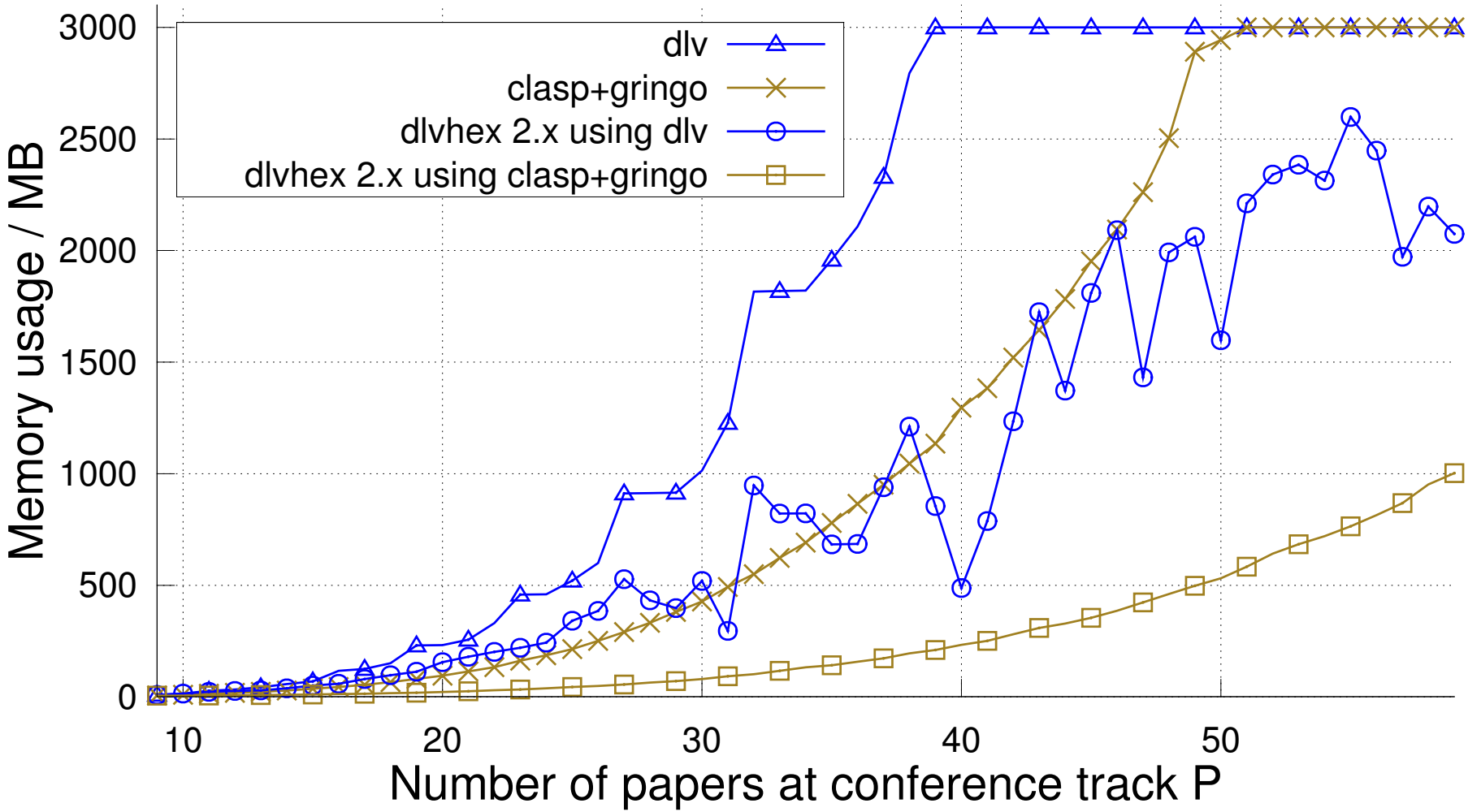
Benchmarks – MCS



Benchmarks – Reviewer Selection ASP



Benchmarks – Reviewer Selection ASP



Conclusion

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/`

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