MCS-IE: Comparison of HEX Rewritings[⋆]

Peter Schüller

Institute of Information Systems
Vienna University of Technology
Favoritenstrasse 11, A-1040 Vienna, Austria
schueller@kr.tuwien.ac.at

May 15, 2010

Abstract. This manuscript gives two possible rewritings from the MCS-IE system input language to HEX programs, and compares them with respect to efficiency of evaluation in dlvhex.

1 Notation

In the following we refer to MCS as defined and explained in [1], we also use notation from [2]. In particular:

- an MCS $M = (C_1, \dots, C_n)$ consists of
- contexts $C_i = (L_i, kb_i, br_i)$ where
- $L = (\mathbf{KB}_L, \mathbf{BS}_L, \mathbf{ACC}_L)$ is a logic (a tuple of two sets and a function, we do not need more details here),
- $kb_i \in \mathbf{KB}_{L_i}$ is a knowledge base (a set),
- br_i is a set of bridge rules of the form

$$(k : s) \leftarrow (c_1 : p_1), \dots, (c_j : p_j), \mathbf{not}\ (c_{j+1} : p_{j+1}), \dots, \mathbf{not}\ (c_m : p_m). \quad (1)$$

  where $1 \leq c_i \leq n$, $1 \leq k \leq n$, $p_i \in \mathbf{BS}_{L_i}$, $s \in \bigcup \mathbf{KB}_{L_i}$,
- IN_i is the set of inputs to a context, which is the set of bridge rule heads s in some rule in br_i,
- OUT_i is the set of output beliefs of a context, which is the set of beliefs p of context i occuring in any bridge rule body as $(i : p)$ or as $\mathbf{not}\ (i : p)$,
- br_M is the set of all bridge rules of an MCS M,
- a diagnosis (D_1, D_2) of some MCS M is s.t. $D_1, D_2 \subseteq br_M$, and
- an inconsistency explanation (E_1, E_2) of some MCS M is s.t. $E_1, E_2 \subseteq br_M$.

HEX programs are an extension of answer set programs with access to external atoms [3, 4].

⋆ This work was supported by the Vienna Science and Technology Fund (WWTF) under grant ICT08-020.

2 Rewriting introduced in [2]

This rewriting is used, if the command-line option `--useKR2010rewriting` is given to dlvhex (and then passed to the MCS-IE plugin).

The MCS is rewritten to the following HEX rules.

$$a_i(p) \vee \bar{a}_i(p). \tag{2}$$

$$\mathit{normal}(r) \vee d_1(r) \vee d_2(r). \tag{3}$$

$$\begin{aligned} b_i(s) \leftarrow\ & \mathit{not}\ d_1(r), a_{c_1}(p_1), \dots, a_{c_j}(p_j), \\ & \mathit{not}\ a_{c_{j+1}}(p_{j+1}), \dots, \mathit{not}\ a_{c_m}(p_m). \end{aligned} \tag{4}$$

$$b_i(s) \leftarrow d_2(r). \tag{5}$$

$$\leftarrow \mathit{not}\ \&\mathit{con\_out}_i[a_i, b_i](). \tag{6}$$

In the following we give an intuition of these rules.

Via (2) we guess for each output belief $p \in OUT_i$ in each context C_i of M whether it is present ($a_i(p)$) or absent ($\bar{a}_i(p)$) in a belief state. This creates an output-projected belief state of M, corresponding to an answer set of the rewritten program.

Additionally we guess a diagnosis via (3), by guessing for each bridge rule $r \in br_M$ whether it is part of D_1, D_2, or not part of one of those sets, encoded as $d_1(r)$, $d_2(r)$, and $\mathit{normal}(r)$, respectively.[1]

For each bridge rule $r \in br_M$, where each one is of the form (1), we have a HEX rule (4) which evaluates the applicability of the bridge rule if it is not in D_1. This yields a set of active ($b_i(s)$) bridge rule heads.

Additionally, for each $r \in br_M$, if the rule is in D_2, its head is also added to the set of active bridge rule heads by (5).

The final step is to kill those answer sets where some context does not accept the guessed output beliefs when presented with the active bridge rule heads as input. This is ensured by one constraint (6) for each context C_i.

Drawbacks. This approach has the practical drawback, that the whole guess is expanded and then each possible equilibrium is checked.

The size of this guess is $3^{|br_M|} \cdot 2^{|OUT_1|} \cdot 2^{|OUT_2|} \dots$, where the first factor is the diagnosis guess and each further factor is the possible output state at each context.

When using dlvhex, this guess is first evaluated using DLV, stored in memory, and then it is successively eliminated by checking external atoms (which only occur in a constraint).

This not only requires a lot of memory for storing all guesses, it also causes a lot of redundant checks: each input/output combination at some context is

[1] This implies, that a bridge rule is never part of both D_1 and D_2. This has no influence on $\subseteq$-minimal sets of diagnoses where $D_1 \cap D_2 = \emptyset$. Furthermore, it is no restriction for obtaining complete sets of non-minimal diagnoses, because each diagnosis (D_1, D_2) can be trivially extended such that all diagnoses are obtained: if (D_1, D_2) is a diagnosis, then $(D_1 \cup D', D_2)$ is a diagnosis for all $D' \subseteq D_2$.

checked multiple times, as it occurs in several answer sets of the rewritten program without the constraint.

In the following, we give an alternative rewriting technique, which eliminates these drawbacks.

3 Alternative Rewriting

This is the default rewriting used by MCS-IE, as it is in most practical applications faster than the rewriting from Section 2.

Again, we first give the rules and then some intuition:

$$\begin{aligned}
&ok_0. &(7)\\
b_i(s) \lor \bar{b}_i(s) &\leftarrow ok_{i-1}. &(8)\\
a_i(p) \lor \bar{a}_i(p) &\leftarrow ok_{i-1}. &(9)\\
ok_i &\leftarrow \&con\_out_i[a_i, b_i](), ok_{i-1}. &(10)\\
&\leftarrow not\ ok_i. &(11)\\
normal(r) \lor d_1(r) \lor d_2(r) &\leftarrow ok_n. &(12)\\
c_i(s) &\leftarrow ok_n, not\ d_1(r), a_{c_1}(p_1), \dots, a_{c_j}(p_j),\\
&\quad\ not\, a_{c_{j+1}}(p_{j+1}), \dots, not\, a_{c_m}(p_m). &(13)\\
c_i(s) &\leftarrow ok_n, d_2(r). &(14)\\
&\leftarrow c_i(s), not\, b_i(s), ok_n. &(15)\\
&\leftarrow not\, c_i(s), b_i(s), ok_n. &(16)
\end{aligned}$$

The basic difference of this rewriting to the previous one is a serialization of guesses, by adding intermediate checks of context semantics. We do not guess all output beliefs and all diagnoses, then evaluate bridge rules and then check contexts. Instead, we guess inputs and output beliefs for the first context, then we check if this is an accepted input/output combination of that context. This we do successively for all context, in order of their indices. After we have a valid input/output combination for all n contexts, we guess a diagnosis, evaluate bridge rules using the previously guessed output beliefs and the diagnosis guess. Finally we kill all answer sets where the bridge rule evaluation does not yield the same inputs we initially guessed for each context.

Concretely, the rewriting works as follows: with (7) we define ok_0 to be true for more comfortable notation.

For each context C_i, ok_i indicates that a valid input/output belief guess exists for that context.

For each context C_i, where ok_{i-1} is true (i.e., we have a successful guess for the previous context), (8) guesses for each bridge rule head $s \in IN_i$ whether it is active ($b_i(s)$) or not active ($\bar{b}_i(s)$), and (9) guesses likewise for each output belief $p \in OUT_i$ whether it is active ($a_i(s)$) or not active ($\bar{a}_i(s)$).

For each context C_i, rule (10) derives ok_i if context C_i accepts the guessed input/output belief combination.

We require that each context accepts the guessed belief set using rules (11).

So far we have the situation, that the HEX evaluation guesses $2^{|IN_i|+|OUT_i|}$ combinations for each context, then immediately verifies them and then continues guessing the next context, until all contexts have a valid input/output combination. In practice, contexts do not accept lots of output combinations for each input combination, therefore the verification step will immediately reject most of the exponentially many guesses.

After the last context has been verified, ok_n is derived. This activates the diagnosis guess (12), and evaluation of bridge rules under the effect of the guessed diagnosis and the guessed output beliefs. This bridge rule evaluation is done by (13) and (14), and the resulting set of active bridge rule heads is represented in $c_i(s)$.

Finally, constraints (15) and (16) check, whether the same atoms are activated for $c_i(s)$ and $b_i(s)$, i.e., whether the guessed active bridge rules correspond to the evaluated active bridge rules.

This rewriting produces the same set of answer sets as the previous one, if projecting to output belief and diagnosis predicates.

Efficiency considerations. No external atoms need to be evaluated, once ok_n has been derived. This allows dlvhex to push the whole diagnosis guess, including verification of bridge rule applicability, into the external DLV instance. Therefore, the whole diagnosis guess can be processed more efficiently, and only answer sets that contain valid diagnosis guesses need to be communicated between dlvhex and DLV.

Comparison of both Rewritings. The second rewriting has a larger worst case space complexity than the first one: if all contexts accept all output beliefs, then a guess of size $\Pi_{1 \leq i \leq n} 2^{|IN_i|+|OUT_i|}$ must be sent to DLV for the diagnosis guess, which then multiplies this guess by $3^{|br_M|}$.

However, such context behavior is not encountered in practice, therefore this rewriting is in practice more efficient than the previous one. Benchmarks indicate a hundredfold increase in speed if no virtual memory swapping is involved, and even higher performance gains for larger input instances which would require swapping with the first rewriting and do not with the second one.

Note that these considerations are oriented toward the current implementation of dlvhex, described in [4].

References

1. Brewka, G., Eiter, T.: Equilibria in heterogeneous nonmonotonic multi-context systems. In: AAAI. (2007) 385–390
2. Eiter, T., Fink, M., Schüller, P., Weinzierl, A.: Finding explanations of inconsistency in nonmonotonic multi-context systems. In: KR. (2010) to appear.
3. Eiter, T., Ianni, G., Schindlauer, R., Tompits, H.: A uniform integration of higher-order reasoning and external evaluations in answer-set programming. In: IJCAI. (2005) 90–96
4. Schindlauer, R.: Answer-set programming for the Semantic Web. PhD thesis, Vienna University of Technology (2006)