Online appendix for the paper

ASP with non-Herbrand Partial Functions: a Language and System for Practical Use

published in Theory and Practice of Logic Programming

MARCELLO BALDUCCINI
Eastman Kodak Company
(*e-mail:* marcello.balduccini@gmail.com)

submitted 10 April 2013; revised 23 May 2013; accepted 23 June 2013

Appendix A Sample of ASP{f} and ASP Encodings from the Case Study

In order to give a sample of the encodings used in the case study, in this section we show key rules for dealing with production estimation. The rules used for dealing with shipping estimation are similar.

As typical, the main input to the system is described by facts. For example, the following ASP{f} facts describe an order, r_1, for 10 items of product p_1 with quoting deadline 13 (e.g. 13 days in the future) and destination l_{10} (of course the declaration needs to occur only once in the program).

#nherb $quantity/1$.
#nherb $product/1$.
#nherb $deadline/1$.
#nherb $destination/1$.

$rfq(r_1)$.
$product(r_1) =_{\#} p_1$.
$quantity(r_1) =_{\#} 10$.
$deadline(r_1) =_{\#} 13$.
$destination(r_1) =_{\#} l_{10}$.

For consistency with the terminology in use in the application domain, orders are identified by relation rfq, which stands for *request for quote*. The ASP encoding of the same information is:

$rfq(r_1)$.
$product(r_1, p_1 1)$.
$quantity(r_1, 10)$.
$deadline(r_1, 13)$.
$destination(r_1, l_{10})$.

Information about the average quantity of a product p on order per day can be specified by means of a fact $daily\_order(p) =_{\#} q$. If not specified, this value is assumed to be 0 in the ASP{f}

encoding by the rule

$$daily\_order(P) =_{\#} 0 \leftarrow product(P), \mathit{not}\ daily\_order(P) \neq_{\#} 0.$$

and in the ASP encoding by the rules:

$$daily\_order(P, 0) \leftarrow product(P), \mathit{not}\ \neg daily\_order(P, 0).$$
$$\neg daily\_order(P, 0) \leftarrow daily\_order(P, Q), Q \neq 0.$$

The determination of the production peaks is centered around the ASP{f} rule (some auxiliary atoms have been omitted here and below to simplify the presentation):

$$\begin{aligned} &at\_peak\_production(RFQ\_ID, D) \leftarrow \\ &\quad rfq(RFQ\_ID), \\ &\quad product(RFQ\_ID) =_{\#} P, \\ &\quad deadline(RFQ\_ID) =_{\#} QD, \\ &\quad D \leq QD, \\ &\quad expected\_on\_order(P, D) =_{\#} max[expected\_on\_order(P, DAY_p) : DAY_p \leq QD]. \end{aligned}$$

The rule informally states that order RFQ_ID is at peak for production on day D if D is no later than the quoting deadline, RFQ_ID is for product P, and the expected quantity of product P on order on day D is equal to the maximum quantity of P expected to be on order on each day from now until the quoting deadline. The corresponding rule in the ASP encoding is:

$$\begin{aligned} &at\_peak\_production(RFQ\_ID, D) \leftarrow \\ &\quad rfq(RFQ\_ID), \\ &\quad product(RFQ\_ID, P), \\ &\quad deadline(RFQ\_ID, QD), \\ &\quad D \leq QD, \\ &\quad expected\_on\_order(P, D, Q), \\ &\quad Q = max[expected\_on\_order(P, DAY_p, Q_p) = Q_p : DAY_p \leq QD]. \end{aligned}$$

Function $expected\_on\_order$ is defined in the ASP{f} encoding by rules such as:

$$\begin{aligned} &expected\_on\_order(P, D) =_{\#} \\ &\qquad \_today\_quantity - \_due\_quantity + M * \_daily\_order \leftarrow \\ &\quad product(RFQ\_ID) =_{\#} P, \\ &\quad current\_time(T), \\ &\quad M = D - T, \\ &\quad daily\_order(P) =_{\#} \_daily\_order, \\ &\quad M <_{\#} avg\_deadline(P), \\ &\quad \_today\_quantity =_{\#} sum[rfq(RFQ\_ID_p) : product(RFQ\_ID_p) =_{\#} P \\ &\qquad = quantity(RFQ\_ID_p)], \\ &\quad \_due\_quantity =_{\#} sum[due\_quantity(P, D') : D' < D]. \end{aligned}$$

This rule intuitively states that the quantity expected to be on order for product P on day D is obtained by subtracting, from the quantity on order now, the quantity due between now and day D, and then adding to this value the average daily orders for$P$ multiplied by the number of days in the period considered. (This formula is motivated by domain-specific considerations.)

The corresponding rule in the ASP encoding is:

$$
\begin{aligned}
&expected\_on\_order(P, D, TODAY\_Q - DUE\_Q + M * DAILY\_ORDER) \leftarrow \\
&\quad product(RFQ\_ID, P), \\
&\quad current\_time(T), \\
&\quad M = D - T, \\
&\quad daily\_order(P, DAILY\_ORDER), \\
&\quad avg\_deadline(P, AVG\_DLINE), \\
&\quad M < AVG\_DLINE, \\
&\quad TODAY\_PAGES = sum[quantity(RFQ\_ID_p, Q) = Q : rfq(RFQ\_ID_p) \\
&\qquad\qquad : product(RFQ\_ID_p, P)], \\
&\quad DUE\_PAGES = sum[due\_quantity(P, D', Q) = Q : D' < D].
\end{aligned}
$$

Appendix B Knowledge Representation in ASP{f}: an Overview

In this section we give an overview of how ASP{f} can be used for some classical knowledge representation tasks. An extended discussion can be found in (Balduccini 2012). More advanced knowledge representation topics are addressed in (Balduccini and Gelfond 2012).

Consider the statements: (1) the value of $f(x)$ is a unless otherwise specified; (2) the value of $f(x)$ is b if $p(x)$ (this example is from (Lifschitz 2011); for simplicity of presentation we use a constant as the argument of function f instead of a variable as in (Lifschitz 2011)). These statements can be encoded in ASP{f} as follows:

$$
\begin{aligned}
&(r_1)\ f(x) = a \leftarrow \textit{not } f(x) \neq a. \\
&(r_2)\ f(x) = b \leftarrow p(x).
\end{aligned}
$$

Rule r_1 encodes the default, and r_2 encodes the exception. The informal reading of r_1 is "if there is no reason to believe that $f(x)$ is different from a, then $f(x)$ must be equal to a".

Extending a common ASP methodology, the choice of value can be encoded in ASP{f} by means of default negation. Consider the statements (again, adapted from (Lifschitz 2011)): (1) the value $f(X)$ is a if $p(X)$; (2) otherwise, the value of $f(X)$ is arbitrary. Let the domain of variable X be given by a relation $dom(X)$, and let the possible values of $f(X)$ be encoded by a relation $val(V)$. A possible ASP{f} encoding of these statements is:

$$
\begin{aligned}
&(r_1)\ f(X) = a \leftarrow p(X),\ dom(X). \\
&(r_2)\ f(X) = V \leftarrow dom(X),\ val(V), \textit{not } p(X),\ \textit{not } f(X) \neq V.
\end{aligned}
$$

Rule r_1 encodes the first statement. Rule r_2 formalizes the arbitrary selection of values for $f(X)$ in the default case. It is important to notice that, although r_2 follows a strategy of formalization of knowledge that is similar to that of ASP, the ASP{f} encoding is more compact than the corresponding ASP one. In fact, the ASP encoding requires the introduction of an extra rule formalizing the fact that $f(x)$ has a unique value:

$$
\begin{aligned}
&(r'_1)\ f'(X) = a \leftarrow\ p(X),\ dom(X). \\
&(r'_2)\ f'(X, V) \leftarrow dom(X),\ val(V),\ \textit{not } p(X),\ \textit{not } \neg f'(X, V). \\
&(r'_3)\ \neg f'(X, V') \leftarrow val(V),\ val(V'),\ V \neq V',\ f'(X, V).
\end{aligned}
$$

The behavior of a dynamic domain consisting of a button b_i, which increments a counter c,

and a button b_r, which resets it, can be encoded in ASP{f} by:

$$(r_1)\ val(c, S+1) = 0 \leftarrow\ pressed(b_r, S).$$
$$(r_2)\ val(c, S+1) = N+1 \leftarrow pressed(b_i, S),\ val(c, S) = N.$$
$$(r_3)\ val(c, S+1) = N \leftarrow val(c, S) = N,\ \textit{not}\ val(c, S+1) \neq val(c, S).$$

Rules r_1 and r_2 are a straightforward encoding of the effect of pressing either button (variable S denotes a time step). Rule r_3 is the ASP{f} encoding of the law of inertia for the value of the counter, and states that the value of c does not change unless it is forced to. For simplicity of presentation, it is instantiated for a particular fluent, but could be as easily written so that it applies to arbitrary fluents from the domain.

References

BALDUCCINI, M. 2012. Answer Set Solving and Non-Herbrand Functions. In *Proceedings of the 14th International Workshop on Non-Monotonic Reasoning (NMR'2012)*, R. Rosati and S. Woltran, Eds.

BALDUCCINI, M. AND GELFOND, M. 2012. Language ASPf with Arithmetic Expressions and Consistency-Restoring Rules. In *ICLP12 Workshop on Answer Set Programming and Other Computing Paradigms (ASPOCP12)*.

LIFSCHITZ, V. 2011. Logic Programs with Intensional Functions (Preliminary Report). In *ICLP11 Workshop on Answer Set Programming and Other Computing Paradigms (ASPOCP11)*.