

1 Program Syntax

1.1 Basic notions

For demonstration purposes, consider the following program Π with aggregates whose syntax is exactly accepted by the solver GROC:

$$uedge(X, Y) : - edge(X, Y). \quad (1)$$

$$uedge(Y, X) : - edge(X, Y). \quad (2)$$

$$\{dhc(X, Y)\} : - uedge(X, Y). \quad (3)$$

$$: - card[dhc(X, Y) : uedge(X, Y)] \geq 2, vtx(Y). \quad (4)$$

$$: - card[dhc(X, Y) : uedge(X, Y)] \geq 2, vtx(X). \quad (5)$$

$$: - vtx(X), \textbf{not} reached(X). \quad (6)$$

$$reached(Y) : - dhc(X, Y), uedge(X, Y), initialvtx(X). \quad (7)$$

$$reached(Y) : - dhc(X, Y), uedge(X, Y), reached(X), \textbf{not} initialvtx(X). \quad (8)$$

$$hc(X, Y) : - dhc(X, Y), edge(X, Y). \quad (9)$$

$$hc(X, Y) : - dhc(Y, X), edge(X, Y). \quad (10)$$

$$: - sum[hc(X, Y) : edgewt(X, Y, Z) = Z] \geq W + 1, maxweight(W). \quad (11)$$

Incidentally, Π is a decision version of the *traveling salesman* program such that we have *vtx*, *edge*, *edgewt*, *initialvtx*, and *maxweight* as its *extensional* predicates, *uedge*, *reached*, and *hc* as the *intensional* predicates, and *dhc* as a *choice* predicate.

Note that rule (3):

$$\{dhc(X, Y)\} : - uedge(X, Y),$$

is the so-called *choice rule*. In this rule, given that *uedge*(*X*, *Y*) holds, we choose arbitrarily whether or not to include *dhc*(*X*, *Y*). In our syntax, we can allow more complicated expressions in the “choice” aggregate such as

$$\{uedge(Y, X) : dhc(X, Y), dhc(X, Y) : reached(Y)\},$$

where each of “*uedge*(*Y*, *X*) : *dhc*(*X*, *Y*)” and “*dhc*(*X*, *Y*) : *reached*(*Y*)” are compound expressions such that ‘:’ acts like a conjunctive operator and are called *aggregate bodies* (we can have as many compounded terms and aggregate bodies as we want). Quite simply, a choice predicate is implicitly inferred from the program by having it declared (in the input instance file) as an intensional predicate but where it does not appear in a *non-aggregate* head of a rule (*dhc* in the case of Π above).

Still in regards to the choice predicates, note that we can also express rule (3) in terms of the *card* aggregate as follows:

$$card[dhc(X, Y)] \geq 0 : - uedge(X, Y), \quad (12)$$

where $card[dhc(X, Y)] \geq 0$ means that there are 0 or more instances of the choice predicate *dhc*(*X*, *Y*) that can include in the answer set. Note that even if we replace rule (3) above by (12), we still have that *dhc* will be inferred from the program as a choice predicate since it is still not mentioned in a *non-aggregate* head of a rule.

As in the case of the “choice aggregate,” we also allow in our syntax the more complicated compounded type of expressions with more than one aggregate bodies, e.g.,

$$card[uedge(Y, X) : dhc(X, Y), dhc(X, Y) : reached(Y)] \geq 0. \quad (13)$$

In fact, we allow this for any of the 5 aggregates we support: *count*, *card*, *sum*, *min*, and *max*. For instance, consider the *sum* aggregate below:

$$sum[hc(X_1, Y_1) : edgewt(X_1, Y_1, Z_1) = Z_1, dhc(X_2, Y_2) : edgewt(X_2, Y_2, Z_2) = Z_2] \geq W. \quad (14)$$

Then we have that (14) is a *sum* aggregate with two aggregate bodies (made up of compounded terms)

$$hc(X_1, Y_1) : edgewt(X_1, Y_1, Z_1) = Z_1$$

and

$$dhc(X_2, Y_2) : edgewt(X_2, Y_2, Z_2) = Z_2.$$

In this case, we take the sum of all those Z_1 and Z_2 that makes the two aggregate bodies $hc(X_1, Y_1) : edgewt(X_1, Y_1, Z_1)$ and $dhc(X_2, Y_2) : edgewt(X_2, Y_2, Z_2)$ true, respectively.

1.2 The left and right comparison operators

In regards to the comparison operators, our current syntax supports aggregates of the following form:¹

$$\text{OP}[\widehat{Body_1}(\vec{z_1}) = n_1, \dots, \widehat{Body_k}(\vec{z_k}) = n_k] \preceq N_1 + \dots + N_l, \quad (15)$$

where:

- $\text{OP} \in \{\text{count}, \text{card}, \text{sum}, \text{min}, \text{max}\};$
- $\preceq \in \{<, \leq, =, \geq, >\};$
- each N_i (for $1 \leq i \leq l$) could either be a number constant (as in 0, 1, 2, ... etc.) or a variables (as in X , Y , or Z);
- each $\widehat{Body_i}(\vec{z_i})$ (for $1 \leq i \leq k$), where $\vec{z_i}$ denotes the tuple of its “free” variables,² are of the form

$$P_1(\vec{x_1}) : \dots : P_m(\vec{x_m}) : \text{not } Q_1(\vec{y_1}) : \dots : \text{not } Q_n(\vec{y_n}) \quad (16)$$

such that ‘:’ in here behaves like the conjunctive operator;

- as in each of the N_i (for $1 \leq i \leq l$) in the comparison expression, each of the n_j (for $1 \leq j \leq k$) can either be a number constant or a variable.

In our syntax, we can drop the n_i (for $1 \leq i \leq l$) terms of the aggregate (as shown in (15)) when op is *count* or *card* ((12) and (13) are examples of this). When the n_i ’s are left in the COUNT or CARD aggregates, the n_i ’s are automatically converted into the number constant ‘1.’

Example 1 In this example, we show the difference between the count and card aggregates. Consider the following aggregate atom

$$\text{OP}[P(X), P(Y)] \geq 3 \quad (17)$$

(i.e., note here that $P(X)$ and $P(Y)$ are two different aggregate bodies), where $\text{OP} \in \{\text{count}, \text{card}\}$. Then grounding (17) on the domain $\{1, 2\}$ produces the aggregate with the corresponding multiset

$$\text{OP}[P(1), P(2), P(1), P(2)] \geq 3. \quad (18)$$

Then in this case, we have that

$$\{P(1), P(2)\} \models \text{count}[P(1), P(2), P(1), P(2)] \geq 3,$$

while

$$\{P(1), P(2)\} \not\models \text{card}[P(1), P(2), P(1), P(2)] \geq 3,$$

since count considers repeated elements while card do not. Thus, the only real difference between count and card is that the former allows for multiplicities while the latter do not. \square

¹Note that the choice aggregate is not included here since it does not involve comparison expressions.

²More about this notion of “free” variables below.

We also allow in our syntax aggregates of the form

$$M_1 + \dots + M_{l+1} \preceq_1 \text{OP}[\widehat{Body_1}(\vec{z_1}) = n_1, \dots, \widehat{Body_k}(\vec{z_k}) = n_k] \preceq_2 N_1 + \dots + N_{l+2} \quad (19)$$

(i.e., note here that \preceq_1 and \preceq_2 could be different) where, as in each of the N_i (for $1 \leq i \leq l_2$), we have that each of the M_j (for $1 \leq j \leq l_1$) are either number constants or variables as well.

Although we allow aggregates with both a left and right comparison expressions, they can only occur in the *head* of a rule, i.e., as in the following rule:

$$1 \leq \text{sum}[\text{edgewt}(X, Y, Z) = Z] \leq 1 : - \text{hc}(X, Y), \quad (20)$$

which is actually equivalent to

$$\text{sum}[\text{edgewt}(X, Y, Z) = Z] = 1 : - \text{hc}(X, Y). \quad (21)$$

But we cannot have something like

$$\text{reached}(Y) : - \text{hc}(X, Y), 1 \leq \text{sum}[\text{edgewt}(X, Y, Z) = Z] \leq 1, \quad (22)$$

since the aggregate atom “ $1 \leq \text{sum}[\text{edgewt}(X, Y, Z) = Z] \leq 1$ ” in the body has both a left and right comparison expression.

In addition to this syntactic restriction, we also require that *each aggregate atom must have at least a right comparison expression!* That is, we cannot have an aggregate atom of the form

$$1 \leq \text{sum}[\text{edgewt}(X, Y, Z) = Z] \quad (23)$$

where the right comparison expression is missing. Note that (23) can simply be expressed as

$$\text{sum}[\text{edgewt}(X, Y, Z) = Z] \geq 1$$

in our syntax.

1.3 Arithmetic comparison atoms

We also allow in our syntax *positive* body atoms (i.e., cannot occur in the *negative* body) of the form

$$N \preceq X_1 + \dots + X_k, \quad (24)$$

where N could either be a number constant or a variable, $\preceq \in \{<, \leq, =, \geq, >\}$, and each of the X_i (for $1 \leq i \leq k$) must be *variables*, i.e., they cannot be number constants. We call this the *arithmetic comparison atoms*. An example of its usage is in the following two rules:

$$\text{hc}(X, Y) : - \text{edgewt}(X, Y, Z_1), \text{edgewt}(Y, X, Z_1), 50 \geq Z_1 + Z_2. \quad (25)$$

and

$$\text{reached}(Y) : - \text{hc}(X, Y), \text{sum}[\text{dwc}(X_1, Y_1) : \text{edgewt}(X_1, Y_2, Z) = Z] \geq W, W = X + Y. \quad (26)$$

2 Global and local variables

In our syntax, we classify the variables in a rule into two kinds: *global* and *local*. By default, all variables that are mentioned elsewhere within the rule and outside of an aggregate atom is inferred in the GROC system as *global*. For instance, in the following rule

$$\begin{aligned} \text{reached}(Y) : - \text{hc}(X, Y), \text{sum}[\text{dwc}(X, V) : \text{edgewt}(U, Y, Z) = Z] \geq W, \\ \text{maxweight}(W), \end{aligned} \quad (27)$$

we have that X , Y , and W are global variables in the aggregate atom

$$\text{sum}[dhc(X, V) : \text{edgewt}(U, Y, Z) = Z] \geq W, \quad (28)$$

since they are mentioned in the other atoms of the rule (27). In particular, it should be noted that any occurrence of a variable in a comparison expression will make it a global variable by default. On the other hand, we have that V , U , and Z are local variables since they are only mentioned within the aggregate atom.

Roughly speaking, we can “loosely” express (27) in terms of classical FOL with aggregates by

$$\begin{aligned} \forall XYW (hc(X, Y) \wedge \text{sum}[ZUV[1] \mid dhc(X, V) : \text{edgewt}(U, Y, Z)] \geq W \wedge \text{maxweight}(W) \\ \rightarrow \text{reached}(Y)), \end{aligned} \quad (29)$$

where $ZUV[1]$ denotes that we are taking the first position of all the collected tuple ZUV (i.e., $ZUV[1] = Z$) that satisfies “ $dhc(X, V) : \text{edgewt}(U, Y, Z)$.”

In addition, we also view the duplicate occurrences of variables within two distinct aggregate body atoms as global. For instance, in the following rule

$$\begin{aligned} \text{reached}(W) : - \text{sum}[\text{edgewt}(X, Y, Z_1) = Z_1, \text{edgewt}(Y, X, Z_2) = Z_2] \geq W, \\ \text{maxweight}(W), \end{aligned} \quad (30)$$

we have that X and Y (along with W) are inferred in the system as global variables and Z_1 and Z_2 as local. Again, intuitively speaking, we can “loosely” represent this in FOL with aggregates as follows:

$$\begin{aligned} \forall XYW (\text{sum}[Z_1, Z_2 \mid \text{edgewt}(X, Y, Z_1), \text{edgewt}(Y, X, Z_2)] \geq W \wedge \text{maxweight}(W) \\ \rightarrow \text{reached}(W)). \end{aligned} \quad (31)$$

If we want to represent (30) where all the variables in the aggregate are local, then we relabel the variables to end up with the following rule:

$$\begin{aligned} \text{reached}(W) : - \text{sum}[\text{edgewt}(X_1, Y_1, Z_1) = Z_1, \text{edgewt}(Y_2, X_2, Z_2) = Z_2] \geq W, \\ \text{maxweight}(W), \end{aligned} \quad (32)$$

such that we can now again “loosely” represent (32) in FOL with aggregate as follows:

$$\begin{aligned} \forall W (\text{sum}[Z_1 X_1 Y_1 X_2 Y_2[1], Z_2 X_1 Y_1 X_2 Y_2[1] \mid \text{edgewt}(X_1, Y_1, Z_1), \text{edgewt}(Y_2, X_2, Z_2)] \geq W \\ \wedge \text{maxweight}(W) \rightarrow \text{reached}(W)) \end{aligned} \quad (33)$$

(note that in this case, W is the only global variable).

As another example, in the rule

$$\begin{aligned} \text{reached}(W_1) : - \text{sum}[\text{edgewt}(X, Y, Z_1) = Z_1] \geq W_1, [\text{edgewt}(Y, X, Z_2) = Z_2] \geq W_2, \\ \text{maxweight}(W_1), \text{maxweight}(W_2), \end{aligned} \quad (34)$$

we have that X and Y (and along with W_1 and W_2) are global variables with Z_1 and Z_2 being local. The reason that X and Y are global in this case is because they are mentioned in two distinct aggregate atoms in the rule. Again, we can “loosely” express this in FOL with aggregate as follows:

$$\begin{aligned} \forall XYW_1 W_2 (\text{sum}[Z_1 \mid \text{edgewt}(X, Y, Z_1)] \geq W_1 \wedge \text{sum}[Z_2 \mid \text{edgewt}(X, Y, Z_2)] \geq W_2 \\ \wedge \text{maxweight}(W_1) \wedge \text{maxweight}(W_2) \rightarrow \text{reached}(W_2)). \end{aligned}$$

It should be noted that although we have some restrictions in our syntax, it is powerful enough to express all the benchmark programs.