AAgg Rule Semantics—General to Specific

Michael Dingess

Aug 27, 2019

1 Gringo Count Semantics

Gringo syntax supports *aggregate atoms* occuring in the body of a rule having the form

$$s_1 \prec_1 \alpha\{\mathbf{t}_1 : \mathbf{L}_1; \ldots; \mathbf{t}_n : \mathbf{L}_n\} \prec_2 s_2 \tag{1}$$

All $\mathbf{t}_i$ and $\mathbf{L}_i$ form *aggregate elements*, which are non-empty tuples of terms and literals, respectively. α is some operation on the **unique** term tuples $\mathbf{t}_i$ whose corresponding condition $\mathbf{L}_i$ holds. The result of applying α is compared by means of the *comparison predicates* $\prec_1$ and $\prec_2$ to the terms s_1 and s_2, respectively. These comparison predicates may be one of $\{<, \leq, =, \neq\}$. One or both of these comparisons can be omitted.[1]

We use $\alpha = \#count$.

$$s_1 \prec_1 \#count\{\mathbf{t}_1 : \mathbf{L}_1; \ldots; \mathbf{t}_n : \mathbf{L}_n\} \prec_2 s_2 \tag{2}$$

The *count* operation then counts the number of **unique** term tuples $\mathbf{t}_i$ whose corresponding condition $\mathbf{L}_i$ holds. The result the count function is compared by the *comparison predicates* $\prec_1$ and $\prec_2$ to the terms s_1 and s_2.

Term tuples, after removing those whose corresponding conditions do not hold, are evaluated as a set by the aggregate operations. Thus emerges the property that only **unique** term tuples are considered by the $\#count$ operation.
Hence, consider the example of two aggregates modified from the Potassco User Guide.[1]

$$\#count\{\ 3\ :\ cost(1,2,3);\ 3\ :\ cost(2,3,3)\ \} = 1 \tag{3}$$

$$\begin{aligned} \#count\{\ 3,1,2\ &:\ cost(1,2,3); \\ 3,2,3\ &:\ cost(2,3,3)\ \} = 2 \end{aligned} \tag{4}$$

Suppose both functions $cost(1,2,3)$ and $cost(2,3,3)$ hold true.
The first aggregate evaluates to 1.
The second aggregate evaluates to 2.
The $\#count$ operation in the first aggregate evaluates the set:
$< 3,3 > \rightarrow < 3 >$.
The $\#count$ operation in the second aggregate evaluates the set:
$< (3,1,2),(3,2,3) >$

2 AAgg Rule Input Forms

AAgg looks for rules of the form:

$$H :- \underset{1 \leq i \leq b}{,} F(\mathbf{X^i}, \mathbf{Y}) \underset{1 \leq i \leq j \leq b}{,} \mathbf{X^i} \neq \mathbf{X^j}, G. \tag{5}$$

where

- H is the head of a rule, possible empty (making the rule a constraint).
- $\mathbf{X^{1..b}}$ are differing variables, all in the same position of F.
- $\mathbf{Y}$ is a comma seperated list of variables, identical in variable and position for all F in the rule.
- F is a predicate function of parity 1 plus the length of $\mathbf{y}$.
- G is the remaining body of the rule, possibly empty.

and

- b is 2 or greater.
- H, G, and $\mathbf{Y}$ have no occurences of variables $\mathbf{X^{1..b}}$.
- The terms $\underset{1 \leq i \leq j \leq b}{,} \mathbf{X^i} \neq \mathbf{X^j}$ may instead be a continuous chain of comparisons: $\underset{1 \leq i \leq b-1}{,} \mathbf{X^i} < \mathbf{X^{i+1}}$ or $\underset{1 \leq i \leq b-1}{,} \mathbf{X^i} > \mathbf{X^{i+1}}$.
- Some other forms logically equivalent to $\underset{1 \leq i \leq j \leq b}{,} \mathbf{X^i} \neq \mathbf{X^j}$ are accepted.

Semantically, these rules state that H holds if

1. All literals in G hold,
2. all functions $F(\mathbf{X^1}, \mathbf{Y}), F(\mathbf{X^{\cdots}}, \mathbf{Y}), F(\mathbf{X^b}, \mathbf{Y})$ hold,
3. and all variables $\mathbf{X^{1..b}}$ are pairwise non-equal.

Stipulations (2.) and (3.) combine to dictate that H holds if

1. All literals in G hold
2. and there are b pairwise non-equal values for $\mathbf{X}$, denoted $\mathbf{x_{1..b}}$, such that $F(\mathbf{x}, \mathbf{Y})$ holds $\forall \mathbf{x_{1..b}}$ for some $\mathbf{Y}$.

Examples

$$:- F(X'), F(X''), X' \neq X''. \tag{6}$$

$$H(5) :- F(X'), F(X''), X' \neq X''. \tag{7}$$

$$:- F(X'), F(X''), X' \neq X'', G(Z). \tag{8}$$

$$H(Z) :- F(X'), F(X''), X' \neq X'', G(Z). \tag{9}$$

$$:- F(X'), F(X''), F(X'''), X' \neq X'', X' \neq X''', X'' \neq X'''. \tag{10}$$

$$:- F(X'), F(X''), F(X'''), X' < X'', X'' < X''', X''. \tag{11}$$

$$:- F(X', Y), F(X'', Y), X' \neq X''. \tag{12}$$

$$:- F(X', Y, Y'), F(X'', Y, Y'), X' \neq X''. \tag{13}$$

3 AAgg Output Forms

AAgg transforms input-format rules into rules of an output format.

3.1 Case: Length of Y=0

If the length of $\mathbf{Y}$ is zero, AAgg outputs the format:

$$H :- b \leq \#count\{F(\mathbf{X}) : F(\mathbf{X})\}, G. \tag{14}$$

where

- Head H, integer b, predicate function F, and body literals G are as above.

- Aggregate element $F(\mathbf{X}) : F(\mathbf{X})$ follows the form **term** : **Literal** as defined above. The first $F(\mathbf{X})$ is a tuple of one term, which in this case is a function. The second $F(\mathbf{X})$ is a literal, which in this case is also a function.

Semantically, these rules state that H holds if

1. All literals in G hold,
2. and b is less than or equal to the number of unique terms $F(X)$ that hold true

i.e. H holds if

1. All literals in G hold,
2. and b is less than or equal to the number of unique ground values for the variable $\mathbf{X}$ for which the function $F(\mathbf{X})$ holds true.

i.e. H holds if

1. All literals in G hold,
2. and there are b unique values of $\mathbf{X}$, denoted $\mathbf{x_{1..b}}$, such that $F(\mathbf{x})$ holds $\forall \mathbf{x_{1..b}}$.

Examples
Rule (6) is transformed into the following...

$$:- 2 \leq \#count\{F(\mathbf{X}) : F(\mathbf{X})\}. \tag{15}$$

Rules (10) and (11) are transformed into the following...

$$:- 3 \leq \#count\{F(\mathbf{X}) : F(\mathbf{X})\}. \tag{16}$$

3.2 Case: Length of Y>0

If the length of $\mathbf{Y}$ is greater than zero, AAgg performs projection on F to project out the variable $\mathbf{X}$. This creates two rules:

$$H :- b \leq \#count\{F(\mathbf{X}, \mathbf{Y}) : F(\mathbf{X}, \mathbf{Y})\}, F'(\mathbf{Y}), G. \tag{17}$$

$$F'(\mathbf{Y}) :- F(\mathbf{X}, \mathbf{Y}). \tag{18}$$

where

- Head H, integer b, predicate function F, variables $\mathbf{Y}$, and body literals G are as above.
- Aggregate element $F(\mathbf{X}, \mathbf{Y}) : F(\mathbf{X}, \mathbf{Y})$ again follows the form **term** : **Literal** as above.
- F' is a new predicate function of parity equal to the number of variables in $\mathbf{Y}$, i.e. equal to the parity of the F minus one.

Semantically, H holds if

1. All literals in G hold,
2. there are b unique lists of values of $\mathbf{X}, \mathbf{Y}$ such that $F(\mathbf{X}, \mathbf{Y})$ holds for all of such lists of values.
3. and $F'(\mathbf{Y})$ holds.

$F'(\mathbf{Y})$ holds for a list of ground values of $\mathbf{Y}$ if

1. There exists a ground value of $\mathbf{X}$ for which $F(\mathbf{X}, \mathbf{Y})$ holds.

i.e. H holds for a list of ground values of $\mathbf{Y}$ if

1. All literals in G hold,
2. there are b unique lists of values of $\mathbf{X}, \mathbf{Y}$ such that $F(\mathbf{X}, \mathbf{Y})$ holds for all of such lists of values.
3. and there exists a ground value of $\mathbf{X}$ for which $F(\mathbf{X}, \mathbf{Y})$ holds.

i.e. H holds for a list of ground values of $\mathbf{Y}$ if

1. All literals in G hold,
2. there are b values of $\mathbf{X}$ such that $F(\mathbf{X}, \mathbf{Y})$ holds for all values of $\mathbf{X}$ for those ground values of $\mathbf{Y}$.
3. and there exists a value of $\mathbf{X}$ for which $F(\mathbf{X}, \mathbf{Y})$ holds.

Note the second item subsumes the third.
Thus H holds for a list of ground values of $\mathbf{Y}$ if

1. All literals in G hold,
2. and there are b values of $\mathbf{X}$ such that $F(\mathbf{X}, \mathbf{Y})$ holds for all values of $\mathbf{X}$ for those ground values of $\mathbf{Y}$.

i.e. H holds if

1. All literals in G hold,
2. and there are b values of $\mathbf{X}$, denoted $\mathbf{x_{1..b}}$, such that $F(\mathbf{X}, \mathbf{Y})$ holds $\forall \mathbf{x_{1..b}}$ for some list of ground values for $\mathbf{Y}$.

3.3 Alternative Output Forms

A number of alternative, logically equivalent output forms are available as well, all derived from the previous form.

Observe that $b \leq F$ is inherently disjoint to $F < b$. Using this fact, we can restate the original aggregate expressing using the negation of its disjointed form. Thus we have our original and negated disjoint form of the aggregate expression as follows:

$$b \leq \#count\{F(\mathbf{X}) : F(\mathbf{X})\} \tag{19}$$

$$not\ \#count\{F(\mathbf{X}) : F(\mathbf{X})\} < b \tag{20}$$

Furthermore, Gringo uses integer-only arithmetic. Observe that, under integer arithmetic, the expression $not\ a < b$ for some integers a and b is equivalent to the conjuctive series: (let commas indicate conjuction)

$$not\ a = -\infty,\ not\ a = -\infty + 1,\ ...,\ not\ a = b - 2,\ not\ a = b - 1$$

Let a be the output of the $\#count$ function defined above, which never outputs a value less than 0. Thus we have that the expression $not\ \#count\{\} < b$ is equivalent to the conjuctive series:

$$not\ \#count\{\} = 0,\ ...,\ not\ \#count\{\} = b - 1$$

Therefore, we see that the aggregate expression (20) is logically equivalent to the following:

$$\begin{aligned} &not\ \#count\{F(\mathbf{X}) : F(\mathbf{X})\} = 0, \\ &not\ \#count\{F(\mathbf{X}) : F(\mathbf{X})\} = 1, \\ &..., \\ &not\ \#count\{F(\mathbf{X}) : F(\mathbf{X})\} = b - 1 \end{aligned} \tag{21}$$

Anonymous Variables
No precise definition for the anonymous variable ('_') is given in *The Potassco Guide*; however they state the following:

"One can view [the anonymous variable] as if a new variable name is invented on each occurrence of '_'."[1].

In our case, ground programs using anonymous variables are always identical to ground programs not using anonymous variables. The anonymous variable grounds to exactly those same values which the regular variable would ground to. This is because H, G, and $\mathbf{Y}$ may have no occurences of $\mathbf{X}$, so $\mathbf{X}$ is not constrained elsewhere in the rule. Hence it is as if $\mathbf{X}$ is some new variable name invented on the fly. Thus we see no impact on performance after grounding.

Is this worth removing from the implementation? If so, is it worth removing from the writeup as well?

Removal would provide freedom to use more clear forms of aggregate expressions. Specifically, if we do not use the anonymous variable,

$$H :- b \leq \#count\{\mathbf{X} : F(\mathbf{X})\}, G. \tag{22}$$

$$H :- b \leq \#count\{F(\mathbf{X}) : F(\mathbf{X})\}, G. \tag{23}$$

(22) may replace (23) and

$$H :- b \leq \#count\{\mathbf{X} : F(\mathbf{X}, \mathbf{Y})\}, F'(\mathbf{Y}), G. \tag{24}$$

$$H :- b \leq \#count\{F(\mathbf{X}, \mathbf{Y}) : F(\mathbf{X}, \mathbf{Y})\}, F'(\mathbf{Y}), G. \tag{25}$$

(24) may replace (25).

Still, we could use these simplified forms in the case when an anonymous variable is not requested by the user, and use the current forms when the anonymous variable is requested by the user.

Note that in (24) using $\mathbf{X} : F(\mathbf{X}, \mathbf{Y})$ is identical to using $\mathbf{X}, \mathbf{Y} : F(\mathbf{X}, \mathbf{Y})$, because the function $F'(Y)$ causes the ground values for $\mathbf{Y}$ to remain static within the rule. In other words, there will exactly as many unique ground terms for $\mathbf{X}, \mathbf{Y}$ as there are for $\mathbf{X}$, with the grounding of $\mathbf{Y}$ remaining static, and only the values for $\mathbf{X}$ differing.

References

[1] Gebser, M.; Kaminski, R.; Kaufmann, B.; Lindauer, M.; Ostrowski, M.; Romero, J.; Schaub, T.; Thiele, S.; 2015. Potassco User Guide.