Математическая логика и логическое программирование

mk.cs.msu.ru → Лекционные курсы
→ Математическая логика и логическое программирование (3-й поток)

Блок 47

Логические программы:
оператор отрицания
SLDNF-резолюция

Лектор:
Подымов Владислав Васильевич
E-mail:
valdus@yandex.ru

ВМК МГУ, 2022/2023, осенний семестр

Оператор отрицания (not)

Оператор отрицания **not** — это особый встроенный оператор, устроенный так:

- **not** — это не предикатный символ, не функциональный символ и не константа
- На месте атома в запросе или в теле правила может быть записано выражение **not**(A),[1] где A — это атом
- В декларативной семантике **not**(A) отвечает формуле $\neg A$ в допущении замкнутости мира

А с операционной семантикой оператора **not** попробуем разобраться отдельно

Чтобы не перегружать материал тем, что относится к общему случаю и представляет в основном теоретический интерес, ограничимся семантикой **not** для стандартной стратегии вычислений

1 В Prolog это записывается как «\+ A»

Оператор отрицания (not)

Хотелось бы устроить вычисление логической программы $\mathcal{P}$ так, чтобы для подцели **not**(A) интерпретатор программ *хотя бы для основных атомов* A проверял соотношение $\Phi_{\mathcal{P}} \models_{cwa} \neg A$, и если это верно, то продолжал вычисление, а иначе констатировал тупик

Проверка этого соотношения для ХЛП $\mathcal{P}$ равносильна проверке того, что не существует ни одного успешного вычисления $\mathcal{P}$

В процессе обоснования теоремы Чёрча был обоснован и тот факт, что задача такой проверки алгоритмически неразрешима (и тем более неразрешима, если разрешить использовать оператор **not**)

Значит, в полной мере и эффективно воплотить допущение замкнутости мира в операционной семантике логических программ не получится

Придётся пойти на компромисс и учесть оператор **not** «достаточно разумно», сохранив эффективность вычислений

SLDNF-резолюция

Правило SLDNF-резолюции (SLD with Negation as Failure) — это аналог правила SLD-резолюции, использующийся для логических программ с отрицанием

Есть несколько формулировок этого правила, и для примера сформулируем ту, которая наиболее приближена к «реальной» интерпретации логических программ согласно стандартной стратегии вычислений

Как и для встроенных предикатов, семантику **not** можно задать как сочетание критерия выполнимости и унификатора:

- Критерий выполнимости **not**(A): строится (обходится) дерево вычислений для запроса $?A$ (вспомогательное дерево), и обход этого дерева вполне неуспешен согласно изложенному далее
- Унификатор: ε

Остальные понятия для вычислений программ (успешное вычисление, результат вычисления, вычислимый ответ, дерево вычислений) вводятся так же, как для «SLD», с заменой «SLD» на «SLDNF»

SLDNF-резолюция

При построении вспомогательного дерева возможны три исхода:

1. Дерево построено, оно конечно, и в нём нет ни одного $\square$
   - По итогам обхода не обнаружено ни одно успешное вычисление
   - В этом случае обход дерева вполне неуспешен
2. В процессе обхода дерева получен $\square$
   - Обнаружены успешное вычисление и вычислимый ответ
   - В этом случае обход дерева успешен
3. Дерево и возникающие вспомогательные деревья обходятся бесконечно, и в строящемся фрагменте дерева нет ни одного $\square$
   - Значит, программа «зациклилась» при проверке вполне-неуспешности дерева
   - В этом случае обход дерева неуспешен, но *не вполне*
   - Соответствующее вычисление программы (для исходного запроса) признаётся бесконечным («сингулярная бесконечность»), и обход исходного дерева и всех остальных поддеревьев немедленно завершается

SLDNF-резолюция

Теорема (о корректности SLDNF-резолюции). Для любой ХЛП с операторами отрицания $\mathcal{P}$ и любого запроса с операторами отрицания $\mathcal{Q}$ верно следующее: результат любого успешного SLDNF-резолютивного вычисления $\mathcal{P}$ на $\mathcal{Q}$ является правильным ответом в допущении замкнутости мира

Можно было бы это доказать, но не хочется перегружать изложение малополезными деталями

А аналогичная теорема о полноте оказывается неверной из-за

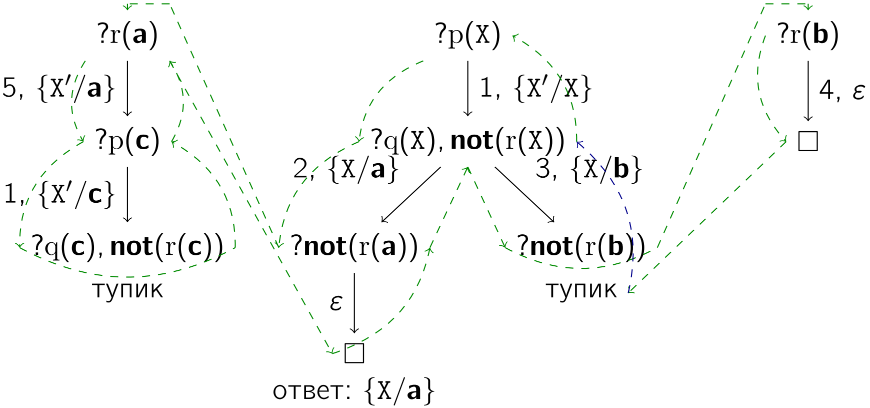
- использования стандартной стратегии вычислений (*этого можно было бы избежать*) и
- возможного появления бесконечных вспомогательных деревьев (*а этого избежать в общем случае не выйдет*)

SLDNF-резолюция

Примеры

$$1:\ p(X) \leftarrow q(X), \mathbf{not}(r(X));$$
$$2: q(\mathbf{a});\quad 3: q(\mathbf{b});$$
$$4: r(\mathbf{b});\quad 5: r(X) \leftarrow p(\mathbf{c});$$

Дерево SLDNF-резолютивных вычислений этой программы для запроса $?p(X)$ и стандартной стратегии вычисления:

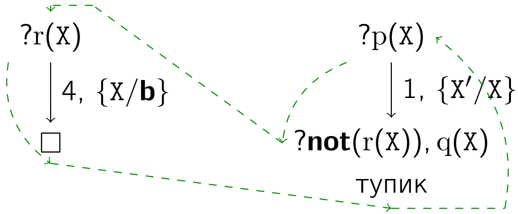


SLDNF-резолюция

Примеры

$$1: \ p(X) \leftarrow \textbf{not}(r(X)), q(X);$$
$$2: q(\mathbf{a}); \quad 3: q(\mathbf{b});$$
$$4: r(\mathbf{b}); \quad 5: r(X) \leftarrow p(\mathbf{c});$$

Дерево SLDNF-резолютивных вычислений этой программы для запроса $?p(X)$ и стандартной стратегии вычисления:



SLDNF-резолюция

Примеры

$$
\begin{aligned}
&1:\ \ \mathrm{p}(\mathtt{X}) \leftarrow \mathrm{q}(\mathtt{X}), \textbf{not}(\mathrm{r}(\mathtt{X}));\\
&2:\mathrm{q}(\mathbf{a});\quad 3:\mathrm{q}(\mathbf{b});\\
&4:\ \ \mathrm{r}(\mathtt{X}) \leftarrow \mathrm{r}(\mathtt{X});
\end{aligned}
$$

Дерево SLDNF-резолютивных вычислений этой программы для запроса $?\mathrm{p}(\mathtt{X})$ и стандартной стратегии вычисления:

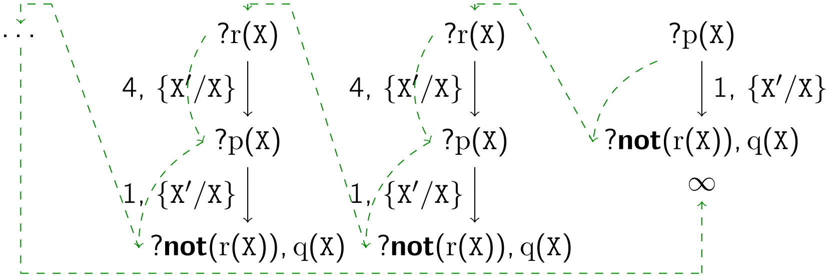
$$
\begin{tikzcd}[column sep=large]
?\mathrm{r}(\mathbf{a}) \arrow[d, "{4,\ \{\mathtt{X}'/\mathbf{a}\}}"] & ?\mathrm{p}(\mathtt{X}) \arrow[d, "{1,\ \{\mathtt{X}'/\mathtt{X}\}}"] \\
?\mathrm{r}(\mathbf{a}) \arrow[d, "{4,\ \{\mathtt{X}'/\mathbf{a}\}}"] & ?\mathrm{q}(\mathtt{X}), \textbf{not}(\mathrm{r}(\mathtt{X})) \arrow[d, "{2,\ \{\mathtt{X}/\mathbf{a}\}}"] \\
?\mathrm{r}(\mathbf{a}) \arrow[d, "{4,\ \{\mathtt{X}'/\mathbf{a}\}}"] & ?\textbf{not}(\mathrm{r}(\mathbf{a})) \\
\ldots & \infty
\end{tikzcd}
$$

SLDNF-резолюция

Примеры

$$
\begin{aligned}
1:&\;\; p(X) \leftarrow \textbf{not}(r(X)), q(X);\\
2:&\; q(\mathbf{a});\quad 3: q(\mathbf{b});\\
4:&\;\; r(X) \leftarrow p(X);
\end{aligned}
$$

Дерево SLDNF-резолютивных вычислений этой программы для запроса $?p(X)$ и стандартной стратегии вычисления:



SLDNF-резолюция

Более «осмысленный» пример

Логическая программа, такая что $\mathrm{p}(\mathtt{X}, \mathtt{L}, \mathtt{M})$ вычисляет в $\mathtt{X}$ произвольный элемент списка $\mathtt{L}$, не содержащийся в списке $\mathtt{M}$:

$$\begin{aligned}
&1 : \mathrm{p}(\mathtt{X}, \mathtt{L}, \mathtt{M}) \leftarrow \mathrm{e}(\mathtt{X}, \mathtt{L}), \mathbf{not}(\mathrm{e}(\mathtt{X}, \mathtt{M}));\\
&2 : \mathrm{e}(\mathtt{X}, \mathtt{X}\textbf{.}\mathtt{L});\\
&3 : \mathrm{e}(\mathtt{X}, \mathtt{Y}\textbf{.}\mathtt{L}) \leftarrow \mathrm{elem}(\mathtt{X}, \mathtt{L});
\end{aligned}$$

Единственный правильный ответ на запрос $?\mathrm{p}(\mathtt{X}, \mathbf{a.b.nil}, \mathbf{b.c.nil})$ в допущении замкнутости мира: $\{\mathtt{X}/\mathbf{a}\}$

SLDNF-резолюция

Более «осмысленный» пример

$$1:\ \mathrm{p}(\mathtt{X},\mathtt{L},\mathtt{M}) \leftarrow \mathrm{e}(\mathtt{X},\mathtt{L}), \mathbf{not}(\mathrm{e}(\mathtt{X},\mathtt{M}));$$

$$2:\mathrm{e}(\mathtt{X},\mathtt{X}.\mathtt{L}); \qquad 3:\mathrm{e}(\mathtt{X},\mathtt{Y}.\mathtt{L}) \leftarrow \mathrm{e}(\mathtt{X},\mathtt{L});$$

Дерево SLDNF-резолютивных вычислений этой программы для $?\mathrm{p}(\mathtt{X}, \mathbf{a.b.nil}, \mathbf{b.c.nil})$ и стандартной стратегии вычисления: