5. Magische Mengen

1. Einleitung (Beispiele, Motivation, Idee)
2. SIP-Strategien
3. Explizite Bindungsmuster
4. Die „Magische Mengen“ Transformation
5. Korrektheit
6. Verbesserungen („Supplementary Predicates“, Rektifizierung)
7. Effizienz der Magischen Mengen (Vergleich SLD-Resolution)

Einleitung (1)

Problem:

- Beispiel (Großeltern von Julia):

$$
\begin{array}{lcl}
\mathrm{elternteil}(X, Y) & \leftarrow & \mathrm{mutter}(X, Y). \\
\mathrm{elternteil}(X, Y) & \leftarrow & \mathrm{vater}(X, Y). \\
\mathrm{gro\text{ß}elternteil}(X, Z) & \leftarrow & \mathrm{elternteil}(X, Y) \wedge \\
 & & \mathrm{elternteil}(Y, Z). \\
\mathrm{answer}(X) & \leftarrow & \mathrm{gro\text{ß}elternteil}(\mathrm{julia}, X).
\end{array}
$$

- Bei der naive/seminaive Bottom-Up Auswertung würden dazu alle Elternteil- und Großelternteil-Beziehungen aller Personen der DB berechnet.

Einleitung (2)

Problem, Forts.:

- Bisher wird die eigentliche Anfrage erst ganz zum Schluß der Berechnung betrachtet — nachdem das komplette minimale Modell des Programms berechnet wurde.
- Also: Berechnung ist nicht zielgerichtet. Viele überflüssige Fakten werden berechnet (nicht relevant für Anfrage).

Lösung:

- Schreibe das Programm so um, daß die Regeln nur „feuern“ können, wenn ihr Ergebnis für die Anfrage relevant ist.
- Dies leistet die „Magische Mengen“ Transformation.

Einleitung (3)

Idee:

- Mache die auftretenden Anfragen und Unteranfragen explizit.
- Dies geschieht in Form sogenannter „magischer“ Prädikate. Zum Beispiel wird „? großelternteil(julia, X)“ codiert als

  m_großelternteil_bf(julia).

- Dies ist nur eine andere Repräsentation. Durch die Angabe des Bindungsmusters kann man Variablen in Fakten vermeiden. Der Name der Variable ist ohnehin nicht wichtig.
- Alle Regeln werden so eingeschränkt, daß sie nur anwendbar sind, wenn eine Anfrage für das Kopfliteral vorliegt.

Einleitung (4)

Beispiel:

$$\begin{array}{lcl}
\mathsf{elternteil}(X, Y) & \leftarrow & \mathsf{m_elternteil_bf}(X) \wedge \\
 & & \mathsf{mutter}(X, Y). \\
\mathsf{elternteil}(X, Y) & \leftarrow & \mathsf{m_elternteil_bf}(X) \wedge \\
 & & \mathsf{vater}(X, Y). \\
\mathsf{gro\ss elternteil}(X, Z) & \leftarrow & \mathsf{m_gro\ss elternteil_bf}(X) \wedge \\
 & & \mathsf{elternteil}(X, Y) \wedge \\
 & & \mathsf{elternteil}(Y, Z). \\
\mathsf{answer}(X) & \leftarrow & \mathsf{true} \wedge \\
 & & \mathsf{gro\ss elternteil}(\mathsf{julia}, X).
\end{array}$$

Einleitung (5)

Beispiel, Forts.:

$$
\begin{array}{lcl}
\text{m\_großelternteil\_bf}(\text{julia}) & \leftarrow & \text{true.} \\
\text{m\_elternteil\_bf}(X) & \leftarrow & \text{m\_großelternteil\_bf}(X). \\
\text{m\_elternteil\_bf}(Y) & \leftarrow & \text{m\_großelternteil\_bf}(X) \wedge \\
 & & \text{elternteil}(X, Y).
\end{array}
$$

Bemerkung:

- Man muß nicht nur die ursprüngliche Anfrage als magisches Fakt angeben, sondern auch Regeln für die Unteranfragen.
- Beispiel: Um die Großeltern von X zu berechnen, muß man erst die Eltern von X berechnen.

Einleitung (6)

Zusammenfassung:

- Eingabe für die „Magische Mengen“ Transformation sind eine Anfrage und ein Programm, das IDB-Prädikate definiert. Bzw. nur ein Programm mit dem speziellen Prädikat „answer“.
- Das „Magische Mengen“ Verfahren produziert ein Programm, das bezüglich der Anfrage äquivalent ist: Das Prädikat „answer” hat im minimalen Modell die gleiche Extension.
- Häufig ist das gesamte minimale Modell aber wesentlich kleiner (es müssen weniger Fakten hergeleitet werden).
- In einem gewissen Sinne werden nur „nötigte Fakten“ berechnet (die Prolog auch berechnen würde).

Einleitung (7)

- Die Äquivalenz gilt unabhängig von den Extensionen der EDB-Prädikate:
  - Man braucht keine DB-Zugriffe während der Transformation.
  - Man kann das transformierte Programm ggf. mehrfach ausführen, auch wenn sich der DB-Zustand geändert hat.
- Die Transformation kann auch das Terminierungsverhalten von Programmen mit eingebauten Prädikaten verbessern.

  Z.B. ist append zunächst eine unendliche Relation. Für eine konkrete Anfrage sind aber nur endlich viele Fakten nötig.

Einleitung (8)

- Es handelt sich um ein „Source Code Level“ Optimierung. Man braucht die darunterliegende „Bottom-Up-Maschine“ nicht zu ändern.
  - Es ist immer gut, Schwierigkeiten zu trennen (Modularität, klar definierte Schnittstellen).
  - Man kann das Verfahren auf hoher Abstraktionsebene verstehen (Herbrand-Modelle statt Datenstrukturen).
  - Allerdings gibt es doch gegenseitige Beeinflussungen. Viele Systeme behandeln die magischen Prädikate speziell.

Einleitung (9)

- Durch das „magischen Mengen“ Verfahren bekommen jetzt auch IDB-Prädikate eine Unterscheidung zwischen Ein- und Ausgabeargumenten.
- Prädikate werden als Prozeduren aufgefaßt:
  - Eingabe: Relation M mit Bindungen für die Eingabeargumente (dies ist gerade die magische Menge).
  - Ausgabe: Relation R für alle Argumente.
    Man kann nicht nur die Ausgabeargumente liefern, weil bei mehreren Tupeln in M die Zuordnung nicht klar wäre.
  - Ist E die Extension des Prädikates, so gilt $R = E \bowtie M$ (wobei alle Attribute von M auch Attribute von E sind).

Einleitung (9)

- Beispiel:

M:

m_elternteil_bf
emil
frida

Prozedur

elternteil_bf

R:

elternteil_bf	
emil	arno
emil	birgit
frida	chris
frida	doris

- Bei rekursiven Prädikaten kann M allerdings nachträglich noch erweitert werden.

SIP-Strategien (1)

Motivation:

- SIP = „Sideways Information Passing“
- Bei der Regelauswertung fließen Information „seitwärts“: von einem früher zu einem später ausgewerteten Rumpfliteral.
- $\text{großelternteil}(X, Z) \leftarrow \text{elternteil}(X, Y) \wedge \text{elternteil}(Y, Z)$.
  - Vom Aufrufer wird die Bindung „X = julia“ vorgegeben.
  - Diese Bindung wird an das erste Rumpfliteral weitergegeben.
  - Dadurch bekommt man Bindungen für Y.
  - Diese werden an das zweite Rumpfliteral weitergegeben.

SIP-Strategien (2)

- Hätte man die Regel

$$\text{großelternteil}(X, Z) \leftarrow \text{elternteil}(X, Y) \wedge \text{elternteil}(Y, Z)$$

  aber mit „? großelternteil(X, arno)“ aufgerufen, so wäre es günstiger, das zweite Rumpfliteral zuerst auszuwerten:
  - Hier könnte die Bindung $Z = \text{arno}$ benutzt werden.
  - Man bekommt dadurch Bindungen für Y, die man dann an das erste Rumpfliteral weitergeben kann.
- Wertet man das erste Rumpfliteral zuerst aus, so würde dies mit dem Bindungsmuster `ff` geschehen. Dann würde man doch die komplette Relation „elternteil“ berechnen.

SIP-Strategien (3)

Definition:

- Gegeben eine Regel $\mathsf{A} \leftarrow \mathsf{B}_1 \wedge \cdots \wedge \mathsf{B}_m$ und ein Bindungsmuster β für pred(A).

- Eine SIP-Strategie legt dann eine Auswertungsreihenfolge für die Rumpfliterale fest, d.h. $\pi\colon \{1, \ldots, m\} \to \{1, \ldots, m\}$ (eine Permutation der Rumpfliterale).

- Außerdem legt eine SIP-Strategie für jedes B_i ein zulässiges Bindungsmuster $\beta_i \in \mathsf{BIND}\big(\mathsf{pred}(\mathsf{B}_i)\big)$ fest, so daß gilt:

$$\mathsf{input\_vars}(\mathsf{B}_i, \beta_i) \subseteq \mathsf{input\_vars}(\mathsf{A}, \beta) \cup \bigcup_{j=1}^{i-1} \mathsf{vars}(\mathsf{B}_{\pi(j)}).$$

SIP-Strategien (4)

Bemerkung:

- Dies ist die gleiche Einschränkung für π und β_i wie in der Definition von „bereichsbeschränkt“.
- Aus einer Auswertungsreihenfolge ergeben sich Bindungsmuster für die Rumpfliterale:
  - Werte für Variablen in „bound“-Argumenten des Kopfes sind bekannt.
  - Werte für Variablen in vorher ausgewerteten Literalen sind bekannt.
  - Alle anderen Variablen haben noch keine Bindungen, führen also zu „free“-Argumentpositionen.

SIP-Strategien (5)

- Normalerweise wählt die SIP-Strategie gerade das speziellste Bindungsmuster, das die Bedingung noch erfüllt (in dem also alle existierenden Bindungen ausgenutzt werden).

- Daher ist die Reihenfolge die eigentliche Entscheidung, die Bindungsmuster dann meist automatisch bestimmt.

  Gibt es jedoch Einschräkungen für die Bindungsmuster, so führt das auch zu Einschränkungen für die Auswertungsreihenfolgen.

- Eine SIP-Strategie kann aber existierende Bindungen ignorieren und ein allgemeineres Bindungsmuster wählen.

  Z.B. weil nicht alle Bindungsmuster implementiert sind, weil man das allgemeinere Bindungsmuster schon an anderer Stelle des Programmes braucht, weil man magische Regeln vereinfachen will.

SIP-Strategien (6)

Aufgabe:

Geben Sie jeweils eine zulässige/gute Auswertungsreihenfolge und entsprechende Bindungsmuster an:

$$\begin{aligned} \text{ware\_nach}(\text{WBez},\ \text{Ort}) \quad \leftarrow \quad & \text{hat\_bestellt}(\text{KNo},\ \text{WNo}) \wedge \\ & \text{kunde\_in}(\text{KNo},\ \text{Name},\ \text{Ort}) \wedge \\ & \text{ware}(\text{WNo},\ \text{WBez}). \end{aligned}$$

- Aufruf: ware_nach(X, 'Hannover').
- Aufruf: ware_nach('Milch', X).
- Aufruf: ware_nach('Milch', 'Hannover').

SIP-Strategien (7)

Beispiele für SIP-Strategien:

- Wähle als nächstes dasjenige unter den auswertbaren Literalen mit den meisten gebundenen Argumenten.
  - Man wählt nacheinander $\pi(1)$, $\pi(2)$, u.s.w.
  - Dabei werden sukzessive mehr Variablen gebunden und damit auch mehr Literale auswertbar.
  - „Auswertbar“ heißt dabei, daß die Bindungen ausreichen, um ein zulässiges Bindungsmuster zu bekommen.
    Z.B. hätte bei $\mathsf{p}(\mathsf{X}, \mathsf{Y}) \leftarrow \mathsf{q}(\mathsf{X}) \wedge \mathsf{X} < \mathsf{Y}$ mit dem Aufruf $\mathsf{p}(\mathsf{X}, 3)$ das zweite Literal die meisten gebundenen Argumente, aber man muß dennoch das erste Literal zuerst auswerten.

SIP-Strategien (8)

Beispiele für SIP-Strategien, Forts.:

- Wähle als nächstes dasjenige unter den auswertbaren Literalen mit den wenigsten freien Argumenten.
  - Dies ist ein Unterschied, weil die Literale unterschiedlich viele Argumente haben können.
  - Beispiel: $p(X_1, X_2) \leftarrow q(X_1, Y) \wedge r(X_1, X_2, Z_1, Z_2)$, Aufruf $p(a, b)$.
- Wähle das am weitesten links stehende auswertbare Literal.
  - D.h. wähle die vom Programmierer vorgegebene Reihenfolge, soweit möglich.

SIP-Strategien (9)

Bemerkung:

- Die SIP-Strategie ist ein wesentlicher Bestandteil des Optimierers eines deduktiven Datenbanksystems.
- Sie sollte u.a. auch folgende Informationen verwenden:
  - Schlüssel
  - Indexe
  - Größe von Relationen
  - Anzahl unterschiedlicher Attributwerte
- Beispiel/Aufgabe: $\mathsf{p}(\mathsf{X}, \mathsf{Z}) \leftarrow \mathsf{q}(\mathsf{X}, \mathsf{Y}_1, \mathsf{Y}_2, \mathsf{Y}_3) \wedge \mathsf{r}(\mathsf{Y}_1, \mathsf{Z})$, Aufruf $\mathsf{p}(\mathsf{a}, \mathsf{b})$, das erste Argument von q und r ist Schlüssel.

SIP-Strategien (10)

Ziele:

- Möglichst kleine Zwischenergebnisse.
- „Fail as early as possible“.
- Teure Prädikate erst aufrufen, wenn billige Tests erfolgreich waren (d.h. teure Prädikate für möglichst wenig Tupel aufrufen). Teure Prädikate können sein:
  - Rekursiv definierte Prädikate.
  - Prädikate, die selbst viele Verbunde und Projektionen mit Duplikateliminierung benötigen.
  - Eingebaute Prädikate mit aufwendigen Berechnungen.

Explizite Bindungsmuster (1)

Ziel:

- Der erste Schritt der „magischen Mengen“-Transformation besteht darin, die Bindungsmuster der IDB-Prädikate explizit zu machen. D.h. wir nennen jeweils das Prädikat p in p_β um.
- Außerdem ordnen wir die Rumpfliterale so um, daß die Auswertung von links nach rechts erfolgt.

Motivation:

- Manchmal muß dasselbe IDB-Prädikat mit unterschiedlichen Bindungsmustern aufgerufen werden.
- Vereinfacht Präsentation der eigentlichen Transformation.

Explizite Bindungsmuster (2)

Definition:

Seien ein Programm P, zulässige Bindungsmuster BIND und eine SIP-Strategie $\mathcal{S}$ gegeben.

- Für jedes Prädikat $\mathsf{p} \in \mathcal{P}_{\mathcal{IDB}}(\mathsf{P})$, $\mathsf{p} \neq \mathsf{answer}$, und jedes Bindungsmuster $\beta \in \mathsf{BIND}(\mathsf{p})$ führen wir ein neues Prädikat $\mathsf{p\_}\beta$ ein.
- Für ein Literal $\mathsf{A} = \mathsf{p}(\mathsf{t}_1, \ldots, \mathsf{t}_n)$ und $\beta \in \mathsf{BIND}(\mathsf{p})$ sei

$$\mathsf{adorn}(\mathsf{A}, \beta) := \begin{cases} \mathsf{p\_}\beta(\mathsf{t}_1, \ldots, \mathsf{t}_n) & \text{falls } \mathsf{p} \in \mathcal{P}_{\mathcal{IDB}}(\mathsf{P}), \\ & \qquad\quad \mathsf{p} \neq \mathsf{answer} \\ \mathsf{p}(\mathsf{t}_1, \ldots, \mathsf{t}_n) & \text{sonst.} \end{cases}$$

Explizite Bindungsmuster (3)

Definition, Forts.:

- Das Program AD(P) enthalte für jede Regel

$$\mathsf{A} \leftarrow \mathsf{B}_1 \wedge \cdots \wedge \mathsf{B}_m \qquad \text{aus IDB(P)}$$

und jedes Bindungsmuster $\beta \in \mathsf{BIND}\big(\mathsf{pred}(\mathsf{A})\big)$ die Regel

$$\begin{aligned} \mathsf{adorn}(\mathsf{A},\, \beta) \quad \leftarrow \quad & \mathsf{adorn}(\mathsf{B}_{\pi(1)},\, \beta_{\pi(1)}) \quad \wedge \cdots \wedge \\ & \mathsf{adorn}(\mathsf{B}_{\pi(m)},\, \beta_{\pi(m)}). \end{aligned}$$

Dabei seien π und $\beta_1, \ldots, \beta_m$ durch die SIP-Strategie $\mathcal{S}$ bestimmt.

Explizite Bindungsmuster (4)

Bemerkung:

- Es ist theoretisch einfacher, jede Regel bezüglich aller möglichen Bindungsmuster für den Kopf zu verarbeiten.
- Die überflüssigen Bindungsmuster fallen heraus, wenn man alle Prädikate wegläßt, von denen „answer“ nicht abhängt.
- Praktisch wird man natürlich gleich nur die nötigen Prädikate und Bindungsmuster betrachten.

  Man hat z.B. eine Menge noch zu bearbeitender Kombinationen, initialisiert mit $\{(\mathsf{answer}, \mathtt{f} \ldots \mathtt{f})\}$. Man nimmt jeweils ein Element heraus und erzeugt die entsprechenden Regeln. Tauchen in den Rümpfen neue (p, β) auf, fügt man sie in die Menge ein.

Explizite Bindungsmuster (5)

- Auch die von der SIP-Strategie gewählten Bindungsmuster für die EDB-Prädikate (inklusive eingebauten Prädikaten) sind nützlich, um zu entscheiden, welche Prozedur / welcher Index verwendet werden soll.
- Die Namen der EDB-Prädikate sind aber im DBS festgelegt, wir können sie nicht umbenennen. IDB-Prädikate ($\neq$ answer) haben dagegen nur Bedeutung innerhalb des Programms.
- Außerdem sind die expliziten Bindungsmuster nur eine Vorbereitung für die „magische Mengen"-Transformation. Diese Transformation würde für EDB-Prädikate keinen Sinn machen, weil ihre Extensionen schon vollständig vorliegen.

Explizite Bindungsmuster (6)

Aufgabe:

Bestimmen Sie AD(P) (soweit für answer relevant):

- $\mathsf{elternteil}(X, Y) \leftarrow \mathsf{mutter}(X, Y).$
  $\mathsf{elternteil}(X, Y) \leftarrow \mathsf{vater}(X, Y).$
  $\mathsf{großelternteil}(X, Z) \leftarrow \mathsf{elternteil}(X, Y) \land \mathsf{elternteil}(Y, Z).$
  $\mathsf{answer}(X) \leftarrow \mathsf{großelternteil}(\mathsf{julia}, X).$

- $\mathsf{append}([\,],\ L,\ L) \leftarrow \mathsf{true}.$
  $\mathsf{append}([H|T],\ L,\ [H|TL]) \leftarrow \mathsf{append}(T,\ L,\ TL).$
  $\mathsf{answer}(X) \leftarrow \mathsf{append}([a],\ [b,\ c],\ X).$

Explizite Bindungsmuster (7)

Aufgabe, Forts.:

- „Same Generation Cousins“:

$$\begin{array}{lll} sg(X, X) & \leftarrow & person(X). \\ sg(X, Y) & \leftarrow & elternteil(X, Xp) \wedge elternteil(Y, Yp) \wedge \\ & & sg(Xp, Yp). \\ answer(X) & \leftarrow & sg(julia, X). \end{array}$$

- Abstraktes Beispiel:

$$\begin{array}{lll} answer(yes) & \leftarrow & p(a, b). \\ p(X, Z) & \leftarrow & q(X, Y) \wedge q(Y, Z). \\ q(X, Y) & \leftarrow & r(a, X, Y). \end{array}$$

Explizite Bindungsmuster (8)

Lemma:

Sei $\mathcal{I}$ das minimale Modell von P und $\mathcal{I}'$ das von $\mathsf{AD(P)} \cup \mathsf{EDB(P)}$.

- Für jedes $\mathsf{p} \in \mathcal{P}_{\mathcal{IDB}}(\mathsf{P})$, $\mathsf{p} \neq \mathsf{answer}$, und jedes $\beta \in \mathsf{BIND(p)}$ gilt:
$$\mathcal{I}'(\mathsf{p\_}\beta) = \mathcal{I}(\mathsf{p}).$$
- Außerdem gilt $\mathcal{I}'(\mathsf{answer}) = \mathcal{I}(\mathsf{answer})$.

D.h. dieser Teil der Transformation hat das minimale Modell nicht wesentlich verändert (nur Prädikate umbenannt und ggf. verdoppelt).

Magische Mengen Transformation (1)

Notation:

- Für ein Literal $\mathsf{A} = \mathsf{p\_}\beta(\mathsf{t}_1, \ldots, \mathsf{t}_n)$ sei

$$\mathsf{magic}[\mathsf{A}] := \mathsf{m\_p\_}\beta(\mathsf{t}_{i_1}, \ldots, \mathsf{t}_{i_k}),$$

  wobei $1 \leq i_1 < \cdots < i_k \leq n$ die Argumentpositionen mit $\beta(i_j) = \mathsf{b}$ seien.

- Für ein Literal $\mathsf{A} = \mathsf{answer}(\mathsf{X}_1, \ldots, \mathsf{X}_n)$ sei $\mathsf{magic}[\mathsf{A}] := \mathsf{true}$.

Beispiel:

- $\mathsf{magic}\big[\mathsf{elternteil\_bf}(\mathsf{X}, \mathsf{Y})\big] := \mathsf{m\_elternteil\_bf}(\mathsf{X})$.

Magische Mengen Transformation (2)

Definition:

Sei P = EDB(P) $\cup$ IDB(P) ein logisches Programm und AD(P) die mit expliziten Bindungsmustern versehene Version von IDB(P). Dann bestehe MAG(P) aus folgenden Regeln:

- Für jede Regel $\mathsf{A} \leftarrow \mathsf{B}_1 \wedge \cdots \wedge \mathsf{B}_m$ in AD(P):

$$\mathsf{A} \leftarrow \mathsf{magic}[\mathsf{A}] \wedge \mathsf{B}_1 \wedge \cdots \wedge \mathsf{B}_m.$$

- Für jede Regel $\mathsf{A} \leftarrow \mathsf{B}_1 \wedge \cdots \wedge \mathsf{B}_m$ in AD(P) und jedes $i \in \{1, \ldots, m\}$, so daß $\mathsf{pred}(\mathsf{B}_i) \in \mathcal{P}_{\mathcal{IDB}}(\mathsf{P})$:

$$\mathsf{magic}[\mathsf{B}_i] \leftarrow \mathsf{magic}[\mathsf{A}] \wedge \mathsf{B}_1 \wedge \cdots \wedge \mathsf{B}_{i-1}.$$

Magische Mengen Transformation (3)

Definition:

- Regeln des Typs $A \leftarrow \mathsf{magic}[\mathsf{A}] \wedge \mathsf{B}_1 \wedge \cdots \wedge \mathsf{B}_m$ heißen „modifizierte Regeln".
- Regeln des Typs $\mathsf{magic}[B_i] \leftarrow \mathsf{magic}[\mathsf{A}] \wedge \mathsf{B}_1 \wedge \cdots \wedge \mathsf{B}_{i-1}$ heißen „magische Regeln".

Definition:

- Ein Fakt (Grundatom), dessen Prädikat die Form $\mathsf{m\_p\_}\beta$ hat, heißt ein „magisches Fakt".
- Alle anderen Fakten heißen „nicht-magische Fakten".

Magische Mengen Transformation (4)

Aufgaben:

- Überprüfen Sie, daß die Definition des „magische Mengen“ Verfahrens für das Großeltern-Beispiel wie erwartet funktioniert.
- Berechnen Sie das Ergebnis der Transformation für append mit Bindungsmuster `bbf`.

Korrektheit (1)

Lemma:

Sei $\mathcal{I}_{\text{AD}}$ das minimale Modell von $\text{AD}(\text{P}) \cup \text{EDB}(\text{P})$ und $\mathcal{I}_{\text{MAG}}$ das minimale Modell von $\text{MAG}(\text{P}) \cup \text{EDB}(\text{P})$. Für alle nicht-magischen Fakten A gilt:

- Ist A in $\mathcal{I}_{\text{MAG}}$, so ist auch $\text{A} \in \mathcal{I}_{\text{AD}}$.
- Ist $\text{A} \in \mathcal{I}_{\text{AD}}$ und $\text{magic}[\text{A}] \in \mathcal{I}_{\text{MAG}}$, so ist auch $\text{A} \in \mathcal{I}_{\text{MAG}}$.

Korrektheit (2)

Satz:

- Seien $\mathcal{I}_{\mathsf{AD}}$ und $\mathcal{I}_{\mathsf{MAG}}$ wie oben, und $\mathcal{I}$ das minimale Modell von $\mathsf{EDB(P)} \cup \mathsf{IDB(P)}$. Dann gilt:

$$\mathcal{I}_{\mathsf{MAG}}(\mathsf{answer}) = \mathcal{I}_{\mathsf{AD}}(\mathsf{answer}) = \mathcal{I}(\mathsf{answer}).$$

- D.h. das transformierte Programm ist äquivalent zum ursprünglichen Programm (liefert dieselben Antworten).

Korrektheit (3)

Satz:

- Ist das Programm P bereichsbeschränkt bezüglich BIND, so ist MAG(P) bereichsbeschränkt bezüglich BIND′, wobei

$$\mathrm{BIND}'(\mathrm{q}) := \begin{cases} \{\mathtt{f}\ldots\mathtt{f}\} & \text{falls q die Form p\_}\beta\text{/m\_p\_}\beta\text{ hat} \\ \mathrm{BIND}(\mathrm{q}) & \text{sonst.} \end{cases}$$

- D.h. das transformierte Programm ist durch Iteration des T_P-Operators auswertbar.

Verbesserungen (1)

Vermeidung wiederholter Berechnungen:

- Beispiel (Regel aus AD(P), alle B_i mit IDB-Prädikaten):

$$\mathsf{A} \leftarrow \mathsf{B}_1 \wedge \mathsf{B}_2 \wedge \mathsf{B}_3.$$

- Ergebnis der „magischen Mengen“ Transformation:

$$\begin{array}{lcl}
\mathsf{magic}[\mathsf{B}_1] & \leftarrow & \mathsf{magic}[\mathsf{A}_1]. \\
\mathsf{magic}[\mathsf{B}_2] & \leftarrow & \mathsf{magic}[\mathsf{A}_1] \wedge \mathsf{B}_1. \\
\mathsf{magic}[\mathsf{B}_3] & \leftarrow & \mathsf{magic}[\mathsf{A}_1] \wedge \mathsf{B}_1 \wedge \mathsf{B}_2. \\
\mathsf{A} & \leftarrow & \mathsf{magic}[\mathsf{A}_1] \wedge \mathsf{B}_1 \wedge \mathsf{B}_2 \wedge \mathsf{B}_3.
\end{array}$$

Verbesserungen (2)

- Lösung: Berechne jeden Join nur einmal, speichere Ergebnis in neuem „supplementary“ Prädikat:

$$
\begin{array}{lcl}
\mathsf{magic}[B_1] & \leftarrow & \mathsf{magic}[A_1]. \\
S_1 & \leftarrow & \mathsf{magic}[A_1] \wedge B_1. \\
\mathsf{magic}[B_2] & \leftarrow & S_1. \\
S_2 & \leftarrow & S_1 \wedge B_2. \\
\mathsf{magic}[B_3] & \leftarrow & S_2. \\
A & \leftarrow & S_2 \wedge B_3.
\end{array}
$$

Verbesserungen (3)

- Die Argumente der S_i sind diejenigen Variablen aus magic[A] $\wedge$ $B_1 \wedge \cdots \wedge B_i$, die noch gebraucht werden, d.h. in $B_{i+1}, \ldots, B_m$, A vorkommen.
- CORAL verwendet dieses Verfahren („Magic Sets with Supplementary Predicates“).
- Es nennt das i-te „Supplementary“-Prädikat der n-ten Regel „sup_n_i“.
- Man kann sich dort das Ergebnis der Transformation in den `*.P.M`-Dateien anschauen.

Verbesserungen (4)

- Beispiel:

$$\mathrm{sg}(X, Y) \leftarrow \mathrm{elternteil}(X, Xp) \wedge \mathrm{sg}(Xp, Yp) \wedge \mathrm{elternteil}(Y, Yp).$$

- Ergebnis (falls elternteil hier EDB-Prädikat):

$$\begin{array}{lcl}
\mathrm{sup_2_1}(X, Xp) & \leftarrow & \mathrm{m_sg_}\mathtt{bf}(X) \wedge \\
 & & \mathrm{elternteil}(X, Xp). \\
\mathrm{m_sg_}\mathtt{bf}(X) & \leftarrow & \mathrm{sup_2_1}(X, Xp). \\
\mathrm{sg_}\mathtt{bf}(X, Y) & \leftarrow & \mathrm{sup_2_1}(X, Xp) \wedge \\
 & & \mathrm{sg_}\mathtt{bf}(Xp, Yp) \wedge \\
 & & \mathrm{elternteil}(Y, Yp).
\end{array}$$

Verbesserungen (5)

Bemerkung/Idee:

- Die magischen Mengen werden bei diesem Verfahren immer direkt aus einem „Supplementary“-Prädikat hergeleitet.
- Man könnte daher versuchen, die magischen Prädikate durch „Supplementary“-Prädikate zu ersetzen.
- Ist ein magisches Prädikat nur durch eine Regel definiert (nur ein Aufruf von p_β), so ist dies nur eine Makro-Expansion.
- Ansonsten müßte man aber Regeln verdoppeln.
- Je nach Anwendung kann es von Vorteil sein, verschiedene „Aufrufe“ eines Prädikates zu unterscheiden.

Verbesserungen (6)

Bindungen an Nicht-Grund-Terme:

- Problem/Beispiel:

$$\mathsf{p}(\mathsf{Y}_1, \mathsf{Y}_2, \mathsf{Y}_3) \leftarrow \mathsf{q}(\mathsf{X}, \mathsf{X}, \mathsf{Y}_1, \mathsf{Y}_2, \mathsf{Y}_3).$$
$$\mathsf{q}(\mathsf{a}, \mathsf{b}, \mathsf{Y}_1, \mathsf{Y}_2, \mathsf{Y}_3) \leftarrow \mathsf{r}(\mathsf{Y}_1) \wedge \mathsf{r}(\mathsf{Y}_2) \wedge \mathsf{r}(\mathsf{Y}_3).$$

- Das bisherige Verfahren unterscheidet nur zwischen „gebundenen“ und „freien“ Argumentpositionen.
- Es würde q mit dem Bindungsmuster `fffff` aufrufen, so daß n^3 Fakten über q hergeleitet werden, wenn es n Fakten über r gibt.

Verbesserungen (7)

- Ein top-down Verfahren wie die SLD-Resolution würde aber sofort erkennen, daß die Regel über q nicht paßt.
- Das Problem ist hier, daß das „magische Mengen“ Verfahren nicht die Information an das aufgerufene Prädikat q weitergegeben hat, daß die ersten beiden Argumente gleich sein müssen.
- Es kann dem aufgerufenen Prädikat nur konkrete Werte für die Argumente übergeben.

Verbesserungen (8)

- Ein Programm heißt rektifiziert gdw.
  - in keinem Rumpfliteral dieselbe ungebundene Variable zweimal vorkommt, und
  - in allen freien Argumentpositionen von Rumpfliteralen nur ungebundene Variablen stehen (keine strukturierten Terme, Konstanten, vorher gebundene Variablen).

    Normalerweise kann die SIP-Strategie existierende Bindungen ignorieren, hier ist das ausgeschlossen.
- Bei den magischen Mengen werden freie Argumentpositionen ja herausprojiziert. Ist das Programm rektifiziert, so geht dadurch keine Information verloren.

Verbesserungen (9)

- Enthält das Programm keine Funktionssymbole, so kann man es in ein äquivalentes, rektifiziertes Programm transformieren.
- Man führt dazu Varianten der Prädikate ein, die an bestimmten Argumentpositionen die gleichen Argumente haben: $\mathsf{p}^{(i_1,\ldots,i_n)}(\mathsf{t}_1,\ldots,\mathsf{t}_k)$ entspreche $\mathsf{p}(\mathsf{t}_{i_1},\ldots,\mathsf{t}_{i_n})$.
  Z.B. $\mathsf{q}^{(1,1,2,3,4)}(\mathsf{X},\mathsf{Y}_1,\mathsf{Y}_2,\mathsf{Y}_3)$ entspricht $\mathsf{q}(\mathsf{X},\mathsf{X},\mathsf{Y}_1,\mathsf{Y}_2,\mathsf{Y}_3)$.
- Man spezialisiere die Regeln einmal für jedes derartige neue Prädikat.
- Zum Beispiel würde die fragliche Regel $\mathsf{q}(\mathsf{a},\mathsf{b},\ldots) \leftarrow \ldots$ bei der Spezialisierung für $\mathsf{q}^{(1,1,2,3,4)}$ gelöscht werden.

Effizienz der magischen Mengen (1)

Beispiel:

Berechnung der von einem Knoten aus erreichbaren Knoten:

$$\begin{array}{lcl} \mathsf{path}(\mathsf{X},\mathsf{Y}) & \leftarrow & \mathsf{link}(\mathsf{X},\mathsf{Y}). \\ \mathsf{path}(\mathsf{X},\mathsf{Z}) & \leftarrow & \mathsf{link}(\mathsf{X},\mathsf{Y}) \wedge \mathsf{path}(\mathsf{Y},\mathsf{Z}). \end{array}$$

? path("http://.../0", X).

Hypertext-Graph:

$$0 \longrightarrow 1 \longrightarrow \cdots \longrightarrow n$$

Effizienz der magischen Mengen (2)

Aufwand der magischen Mengen:

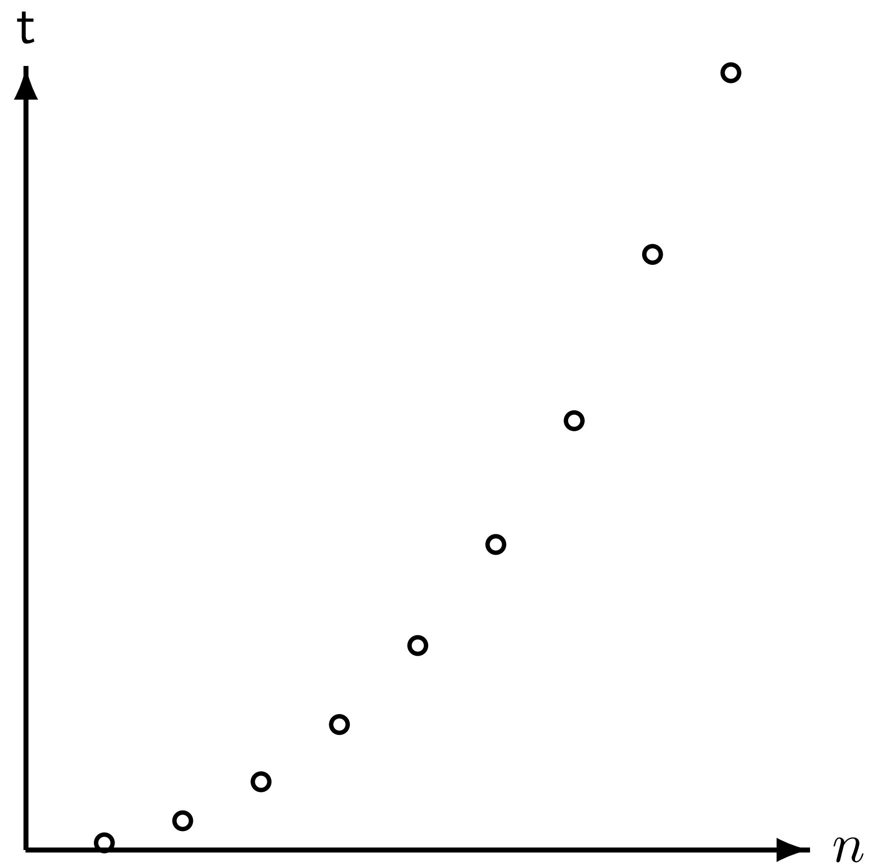
- Die Anfrage path$(0, \mathsf{X})$ ruft rekursiv path$(1, \mathsf{X})$ auf, u.s.w.
- Schließlich werden alle Fakten path(i, j) berechnet.
- Herleitbare Fakten:

m_path_bf(i)	für	$0 \leq i \leq n$
path(i, j)	für	$0 \leq i < j \leq n$

- Anzahl: $(n + 2)(n + 1)/2$,
  also Komplexität mindestens quadratisch.
- Dieses Beispiel sollte eine lineare Komplexität haben!

Effizienz der magischen Mengen (3)

Gemessene Laufzeit (CORAL):

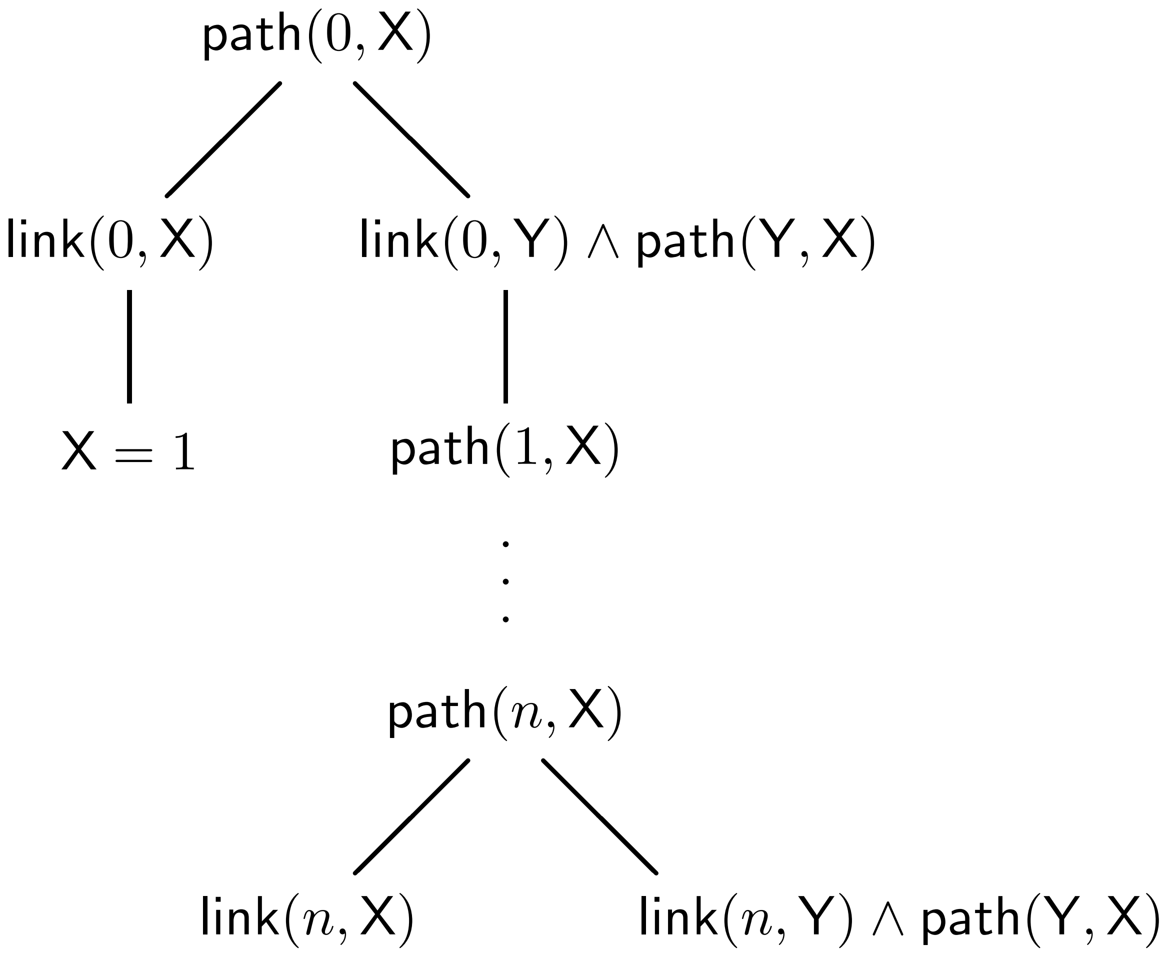


Effizienz der magischen Mengen (4)

SLD-Resolution (Prolog):

- Baut einen Baum von Anfragen auf (siehe nächste Folie).
- Wurzel ist die gegebene Anfrage path$(0, \mathsf{X})$.
- Nun wird versucht, diese Anfrage mit Hilfe der Regeln herzuleiten.
- Zum Beispiel würde path$(0, \mathsf{X})$ gelten, wenn link$(0, \mathsf{X})$ bewiesen werden könnte (Unteranfrage $\equiv$ Nachfolgeknoten).
- Enthält eine Anfrage mehrere Literale, so werden die Regeln zunächst auf ein „ausgewähltes“ Literal angewendet (normalerweise das erste).

Effizienz der magischen Mengen (5)



Effizienz der magischen Mengen (6)

Komplexität der SLD-Resolution:

- Anzahl Knoten im SLD-Baum: $4n + 3$.
- Knoten bestehen aus maximal 2 Literalen.
- Zugriffszeit auf Relation link: $\mathrm{O}\big(\log(n)\big)$.
- Kosten insgesamt: $\mathrm{O}\big(n * \log(n)\big)$.
- Könnte man link_`bf` in $\mathrm{O}(1)$ Zeit auswerten (Hashtabelle), so wäre die Gesamtlaufzeit tatsächlich linear.

Effizienz der magischen Mengen (7)

Satz:

- Sei P ein rektifiziertes Programm.
- Dann gibt es zu jedem magischen Fakt m_p_β($c_1, \ldots, c_k$), das aus MAG(P) $\cup$ EDB(P) herleitbar ist, einen Knoten im SLD-Baum mit ausgewähltem Literal A, so daß

$$\text{magic}[A] = \text{m\_p\_}\beta(c_1, \ldots, c_k).$$

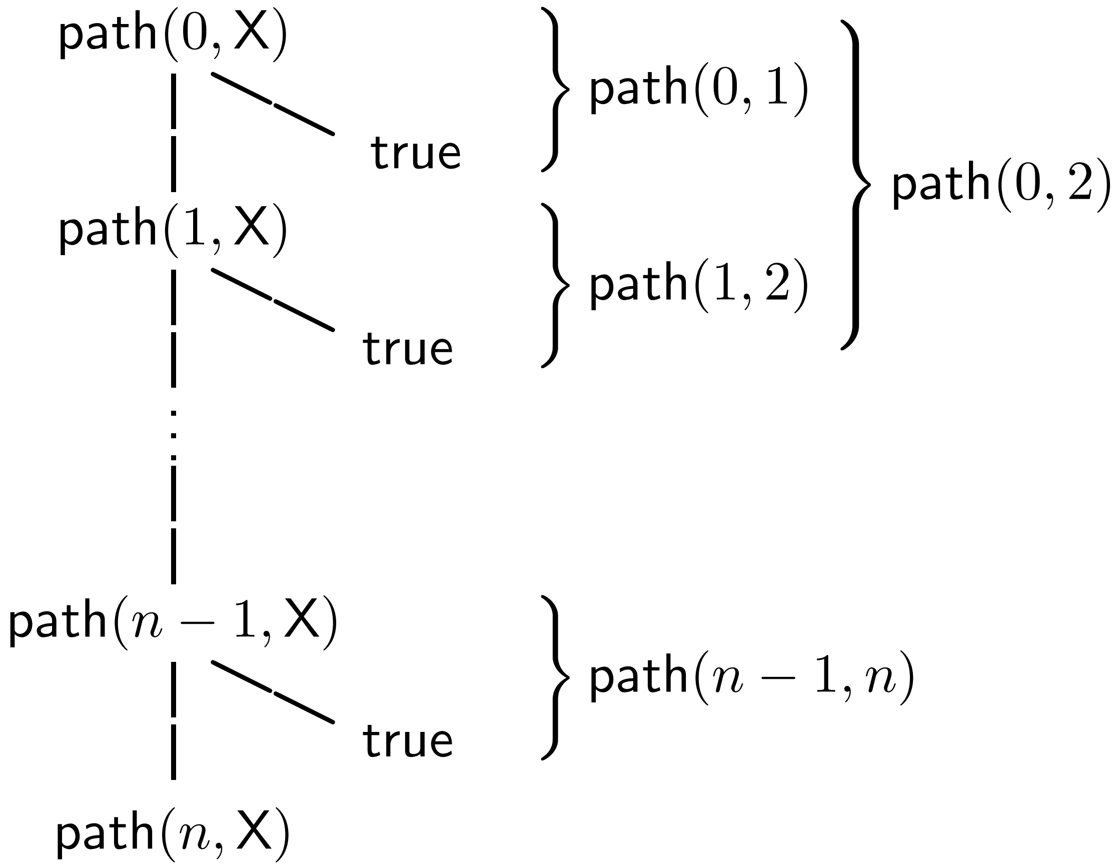
- Im Beispiel: Zu jedem m_path_bf(i) gibt es einen Knoten path(i, X).
- D.h. magische Fakten entsprechen (Unter-)Anfragen im SLD-Baum.

Effizienz der magischen Mengen (8)

Satz:

- Sei P ein rektifiziertes Programm.
- Jedes nicht-magische Fakt, das aus $\mathrm{MAG(P)} \cup \mathrm{EDB(P)}$ herleitbar ist, wird im SLD-Baum als „Lemma“ bewiesen.
- $\mathrm{A}\theta$ wird im SLD-Baum als Lemma bewiesen, wenn ein Knoten $\mathrm{A} \wedge \mathrm{B}_1 \wedge \cdots \wedge \mathrm{B}_n$ einen Nachfolgeknoten (eventuell über viele Zwischenschritte) $(\mathrm{B}_1 \wedge \cdots \wedge \mathrm{B}_n)\theta$ hat, wobei θ die Komposition der in der Zwischenzeit angewendeten Substitution (Unifikatoren) ist.

Effizienz der magischen Mengen (9)



Effizienz der magischen Mengen (10)

Ursache der Probleme:

- Mit einer linearen Anzahl Knoten kann implizit eine quadratische Anzahl Lemmata bewiesen werden.
- Das „magische Mengen“ Verfahren speichert diese Lemmata explizit ab.
- Dieses Problem tritt nur bei Endrekursionen auf.
- Ansonsten gilt: Die Anzahl der anwendbaren Regelinstanzen im magisch transformierten Programm ist

$$O(\text{Knoten des SLD-Baums}).$$

Effizienz der magischen Mengen (11)

- Enthält das Programm keine Funktionssymbole und keine eingebauten Prädikate, so terminiert die Bottom-Up Auswertung und benötigt nur polynomielle Zeit in der Größe der DB (EDB-Fakten).
- Die SLD-Resolution muß nicht terminieren. Selbst wenn sie terminiert, kann sie exponentielle Zeit benötigen.
- Es gibt Varianten der SLD-Resolution, die eine Tabellierung verwenden, um Endlosschleifen und Doppelberechnungen zu vermeiden (z.B. im XSB-System).
- Diese haben aber mit Endrekursionen die gleichen Probleme. Tatsächlich ist dieses Verfahren isomorph zu den „Magic Sets“.

Effizienz der magischen Mengen (12)

Schlußbemerkung:

- Die Literatur ist voll von Verbesserungen des „Magic Set“ Verfahrens.
- Darunter auch die SLDMagic-Methode von Brass, die die Vorteile von SLD-Resolution und Magischen Mengen verbindet.
- Außerdem wird damit noch ein weiteres Problem der magischen Mengen abgestellt: Sie können ein nichtrekursives Eingabe-Programm in ein rekursives Ausgabeprogramm überführen (siehe großelternteil-Beispiel).