Antwortmengenprogrammierung in der Praxis - eine Fallstudie

Petra Schwaiger

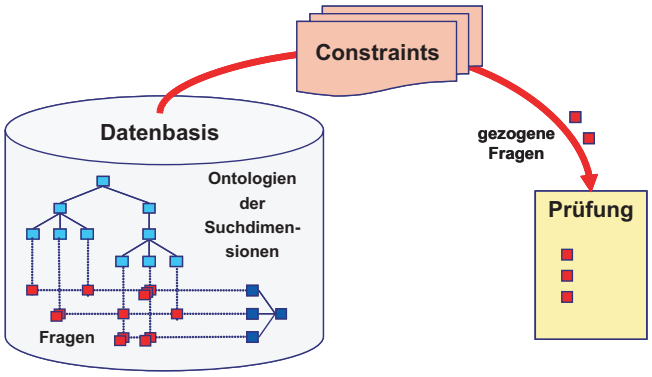
Universität Passau, Fakultät für Mathematik und Informatik
D-94030 Passau, Germany
schwaige@im.uni-passau.de

Zusammenfassung

Die Bedeutung des E-learning zeigt sich unter anderem in der zunehmenden Verwendung von Prüfungsmanagementsystemen, welche unter anderem zum Zwecke der Zusammenstellung von Tests eingesetzt werden. Das Problem der Auswahl von Fragen unter vorgegebenen Randbedingungen ist in der Praxis oftmals nur unzureichend gelöst. In dieser Arbeit stellen wir hiervon eine umfassende Lösung in Smodels vor.

1 Einleitung

In der industriellen Praxis gewinnen verschiedene Aspekte des E-learning an Bedeutung. So erfahren Aus- und Fortbildungsabteilungen von Unternehmen durch Prüfungsmanagementsysteme - wie zum Beispiel der IFIS Assessment.Suite, welche vom Berufsbildungswerk der Bausparkassen und von der Deutschen Bahn AG eingesetzt wird - zunehmende Unterstützung in der Zusammenstellung von Tests.



Bei der Prüfungsgenerierung wird aus einem vorhandenen Datenpool eine Auswahl von Fragen getroffen. In der Regel werden die Fragen durch Metadaten aus verschiedenen Suchdimensionen, wie z. B. Thema, Schwierigkeitsgrad, Frageart und Bearbeitungsdauer, beschrieben. Möglicherweise liegen einzelne Dimensionen als Ontologie, z. B. in Form einer Themenhierarchie, vor. Denkbar sind auch n:m-Assoziationen, was der Fall wäre, wenn eine Frage mehreren Themen zugeordnet ist. Typischerweise sind an die Zusammensetzung der Fragen Bedingungen (Constraints) geknüpft. Diese können in Form von Zahlenbeschränkungen repräsentiert sein. So könnte eine maximale Anzahl von Fragen zu einem bestimmten Thema oder eine Mindestbearbeitungsdauer gefordert sein.

In den industriellen Lösungen ist das Problem der dynamischen Generierung von Prüfungen aus einem vorhandenen Fragenpool unter vorgegebenen Randbedingungen meist nur mit Einschränkungen - und

damit nicht zufriedenstellend - modelliert bzw. gelöst. Je nach Lösungsansatz bereiten die Zahlenrestriktionen, die hierarchischen Strukturen und die damit verbundenen transitiven Anforderungen, die Mehrdimensionalität des Lösungsraums und n:m-Beziehungen - bzw. deren Kombination - Schwierigkeiten.

Smodels ([SN01], [NS00]) stellt gegenüber den normalen logischen Programmen unter anderem eine Erweiterung um Kardinalitäts- und Gewichtsconstraints dar. Mit dem Zugewinn an Semantik lässt sich gerade eben beschriebenes Problem relativ einfach und umfassend lösen.

Ziel unserer Arbeit ist die Präsentation der Lösung des in der Praxis interessanten und relevanten Problems der "Prüfungsfragengenerierung unter Constraints" unter Zuhilfenahme von Smodels.

2 Erläuterungen zur verwendeten Sprache

Die in Smodels verwendete deklarative Sprache ist dem Answer Set Programming ([Lif]) zuzurechnen.

2.1 Grundlagen des Answer Set Programming (ASP)

Ausgangspunkt für ASP-Systeme ist die Implementierung der stabilen Modellsemantik von normalen (logischen) Programmen ([GL88]) mit Regeln der Form $H \leftarrow A_1, ..., A_n, not\, B_1, ..., not\, B_m$ mit atomaren Formeln (Atomen) H, A_1, ..., A_n, B_1, ... B_m ($n, m \geq 0$). Als Literal bezeichnet man ein Atom A (positives Literal) oder dessen Negation $not\, A$ (negatives Literal). Unter not wird hier die Defaultnegation verstanden. In manchen Implementierungen ist zusätzlich zu dieser auch die klassische Negation ($\neg A$) erlaubt.

Typischerweise haben ASP-Programme mehrere stabile Modelle (Antwortmengen), welche die verschiedenen Lösungen des durch das Programm repräsentierten Problems darstellen.

2.2 Erweiterungen

Durch zusätzlich erlaubte Anzahlconstraint-, Gewichtsconstraint- und Bedingungsliterale lassen sich in Smodels Erweiterungen der oben beschriebenen Basisregeln - sogenannte allgemeine Gewichtsconstraint-Regeln - formulieren ([NS00, Syr03]). Diese haben die Gestalt $C_0 \leftarrow C_1, ..., C_n$ mit $n \geq 0$ und Gewichtsconstraints (oder als Sonderfall Basisliterale, Anzahlconstraints) $C_i, 0 \leq i \leq n$. Die zugehörige Semantik stellt eine Erweiterung der stabilen Modellsemantik dar.

Möchte man ausdrücken, dass aus den vorhandenen 4 Fragen "`frage(fi).`" ($1 \leq i \leq 4$) mindestens 1 und höchstens 3 Fragen auszuwählen sind (Prädikat `gezFrage`), so kann dies durch

```
1 {gezFrage(f1), gezFrage(f2), gezFrage(f3), gezFrage(f4)} 3.
```

geschehen. Unter Zuhilfenahme eines Bedingungsliterals (Bedingung: `frage(X)`) lässt sich dies noch wie folgt kürzer ausdrücken:

```
1 {gezFrage(X) : frage(X)} 3.
```

In analoger Weise können Bedingungen an die Summe der Gewichte der Literale (Default: 1) in einem Gewichtsconstraintliteral formuliert werden. Im Folgenden erfolgt nach der Zuordnung des Gewichts die "Ziehung" von Literalen mit dem Prädikat `gezFrageDauer` derart, dass die Summe der Gewichte der vorkommenden Literale einen Wert zwischen 5 und 13 einnimmt:

```
#weight gezFrageDauer(X,Y) = Y.
5 [gezFrageDauer(X,Y) : frageDauer(X,Y)] 13.
```

Um Variablen zulassen zu können und trotzdem die Entscheidbarkeit zu gewährleisten, sind in Smodels nur sogenannte ω-beschränkte Regeln erlaubt, was bedeutet, dass jede Variable der Regel in einem positiven Literal - genauer einem sogenannten positiven Domänenliteral (zur Einschränkung der möglichen Werte der Variablen) - des Rumpfes vorkommen muss.

3 Lösungsansätze in Smodels

3.1 Problemausprägung 1: Eine hierarchische Suchdimension

Die in dem Beispiel-Datenpool vorhandenen 1000 Fragen sind nach Lernzielen (und indirekt Themen) klassifiziert. Genauer gilt, dass jede Frage genau einem Lernziel und jedes Lernziel genau einem Thema zugeordnet ist. An eine Prüfung mit einer vorgegebenen Anzahl von n ($1 \leq n \leq 1000$) Fragen ist die Bedingung geknüpft, dass sie möglichst viele verschiedene Themen und Lernziele abdecken soll.

Durch Fakten wie

```
thema(thA). thema(thB). ...
lernziel(lza). lernziel(lzb). ...
frage(f1). frage(f2). ...
themaLernziel(thA,lza). themaLernziel(thA,lzb). ...
lernzielFrage(lza,f1). lernzielFrage(lza,f2). ...
```

lässt sich die Datenbasis zusammen mit der hierarchischen Struktur von Thema - Lernziel - Frage modellieren. Die indirekte Zuordnung der Fragen zu den Themen erhält man mit der Regel

```
themaFrage(X,Y) :- themaLernziel(X,Z), lernzielFrage(Z,Y).
```

Die Anzahl der zu ziehenden Fragen ist dem Programm als Konstante `n` zu übergeben und kann dort durch das Fakt "`anzGezFragen(n).`" "verfügbar" gemacht werden. Durch die Regel

```
Y {gezFrage(X) : frage(X)} Y :- anzGezFragen(Y).
```

wird das Ziehen der Fragen repräsentiert. Die gezogenen Lernziele und Themen können anhand von

```
gezLernziel(X) :- lernzielFrage(X,Y), gezFrage(Y).
gezThema(X) :- themaLernziel(X,Y), gezLernziel(Y).
```

bestimmt werden. Schließlich ist die Bedingung, dass möglichst viele verschiedene Themen in der Ziehung vertreten sein - also "gezogen" werden - müssen, auszudrücken. Sind mehr Themen als zu ziehende Fragen vorhanden, so stimmt die Anzahl der zu ziehenden Themen mit der Anzahl der zu ziehenden Fragen überein. Im anderen Fall müssen aufgrund des Mangels an Auswahl alle Themen gezogen werden. In einem ersten Schritt wird die Anzahl der vorhandenen Themen gezählt:

```
anz(0..1000).
anzThemen(Y) :- Y {thema(X) : thema(X)} Y, anz(Y).
```

Hierbei wird das Prädikat `anz` aufgrund der geforderten ω-Beschränktheit benötigt; "`anz(0..1000).`" ist eine Abkürzung für die 1001 Regeln "`anz(i).`" ($0 \leq i \leq 1000$). In einem zweiten Schritt kann durch

```
anzGezThemen(X) :- anzGezFragen(X),anzThemen(Y),anz(Y), X <= Y.
anzGezThemen(Y) :- anzGezFragen(X),anzThemen(Y),anz(Y), X > Y.
```

die Anzahl der zu ziehenden Themen bestimmt werden. Im letzten Schritt erfolgt die Formulierung der Zahleneinschränkung:

```
:- {gezThema(X) : thema(X)} Y-1, anzGezThemen(Y).
:- Y+1 {gezThema(X) : thema(X)}, anzGezThemen(Y).
```

Der leere Regelkopf deutet $\perp$ oder $false$ an, was heißt, dass in einem Modell der Regelrumpf nicht erfüllt sein darf. Analog ist die entsprechende Bedingung an die zu ziehenden Lernziele zu modellieren.

3.2 Problemausprägung 2: Zwei Suchdimensionen

Das Beispiel 3.1 wird dahingehend erweitert, dass die Fragen auch in ihrer Fragenart (Single Choice oder Pick and Place) unterschieden werden. Als Randbedingung ist eine Mindestanzahl von Single-Choice-Fragen gefordert.

Die Datenbasis ist durch Fakten wie "`frageArt(f1,sc). frageArt(f2,pp). ...`" erweitert. Analog zu `n` wird auch die Anzahl der zu ziehenden Single Choice Fragen `n_sc` dem System übergeben und durch "`anzGezFragenSc(n_sc).`" repräsentiert. Schließlich sind nur noch die Regeln

```
gezFrageArt(X,Y) :- gezFrage(X), frageArt(X,Y).
:- {gezFrageArt(X,sc) : frage(X)} Y-1, anzGezFragenSc(Y).
```

anzufügen. Die Mehrdimensionalität des Suchraumes stellt insofern keine Schwierigkeit an die Modellierung dar, da die Bedingungen an die zweite Dimension (Frageart) analog zu denen der ersten und vollig unabhängig von diesen formuliert werden können. Dies verdient insofern eine Erwähnung, da manche "Ziehungsalgorithmen" diese Unabhängigkeit nicht gewährleisten und somit den Lösungsraum ungewollt einschränken.

3.3 Problemausprägung 3: n:m-Assoziation

Nehmen wir an, dass die Fragen in unserem Beispiel-Datenpool ausschließlich und direkt nach drei verschiedenen Themen (thA, thB, thC) klassifiziert sind. Die Beziehung Frage - Thema ist vom Typ n:m, so dass eine Frage mehreren Themen zugeordnet sein kann. Eine Prüfung mit einer vorgegebenen Anzahl von n ($1 \leq n \leq 1000$) Fragen soll $n_thA \leq n$ Fragen zu Thema A und $n_thB \leq n$ Fragen zu Thema B enthalten.

Die Datenbasis und die Ziehung kann analog zu obigen Beispielen modelliert werden. Problem hierbei ist, dass eine gezogene Frage nicht mehrfach gezählt werden darf. Wird beispielsweise die Frage $f1$, welche den Themen thA und thB zugeordnet ist, gezogen, so darf sie bei der Zählung der gezogenen Fragen in den einzelnen Themenbereichen nur für eines der beiden Themen gewertet werden. Dies erreicht man dadurch, dass "während" des Fragen-Ziehens eine Entscheidung für ein mögliches zulässiges Thema vorgenommen wird. Dies kann durch

```
gezA(X) :- gezFrage(X),frageThema(X,thA),not gezB(X),not gezC(X).
gezB(X) :- gezFrage(X),frageThema(X,thB),not gezA(X),not gezC(X).
gezC(X) :- gezFrage(X),frageThema(X,thC),not gezA(X),not gezB(X).
```

oder alternativ durch

```
1{gezFrageGezThema(X,Y):frageThema(X,Y)}1 :- gezFrage(X),frage(X).
```

nachgebildet werden. Die Constraints bezogen auf die Anzahl der Fragen in den einzelnen Themen sind in entsprechender Weise zu formulieren.

3.4 Problemausprägung 4: Bedingung nicht über Fragenanzahl, sondern einer anderen numerischen Eigenschaft

Dieser Fall wird hier nur kurz skizziert: Im Gegensatz zu obigen Beispielen sind die Fragen mit einer Bearbeitungsdauer ("`frageDauer(X,Y).`") versehen und die Prüfung durch eine vorgegebene Gesamtbearbeitungsdauer ("`dauer_ges(t).`") bestimmt. Problem hierbei ist, dass nicht von vornherein die Anzahl der zu ziehenden Fragen klar ist. Eine Lösung lässt sich dadurch realisieren, dass die entsprechenden Fragen-Prädikate mit der entsprechenden Fragedauer gewichtet werden und die Ziehung über die Summe der Gewichte erfolgt, wie im Folgenden angedeutet ist:

```
#weight gezFrageDauer(X,Y) = Y.
Z [gezFrageDauer(X,Y) : frageDauer(X,Y)] Z :- dauer_ges(Z).
```

4 Diskussion

Obige kurze Liste von Problemausprägungen lässt sich sinnvoll und problemlos erweitern. So kann man zum Beispiel durch eine geeignete Modellierung erreichen, dass, falls inkonsistente Anforderungen an die zu generierende Prüfung gestellt werden, ein entsprechender Hinweis mit möglicher Ursache vom System ausgegeben wird. Smodels scheint ein geeignetes Mittel zu sein, vielerlei Anforderungen, welche im Rahmen der "Generierung von Prüfungen unter Constraints" auftreten, zu modellieren. Im Sinne der Ziele der Working Group on Answer Set Programming (WASP) ist hiermit also ein in der Industrie auftretender Anwendungsfall gefunden, welcher den Nutzen der ASP-Technologie zeigt.

Allerdings erinnern natürlicherweise manche gelöste (Teil-)Probleme an Aufgaben, die auch im Constraint-Programming (vgl. auch [Nie99]) oder im Operations Research gelöst werden. Eine genauere Abgrenzung zu diesen Bereichen gehört zu den nächsten Aufgaben. Desweiteren soll eine Verbindung von ontologischem Reasoning - die einzelnen Suchdimensionen sind oftmals als Ontologien gegeben - und Smodels untersucht werden.

Testläufe haben ergeben, dass der gezeigte Ansatz mit realen Daten sinnvolle Ergebnisse liefert. Eine genauere Untersuchung bzgl. Effizienz und Skalierbarkeit ist jedoch noch erforderlich.

5 Danksagung

Besonderer Dank gilt Prof. Dr. Burkhard Freitag für die wertvollen Anregungen und Ratschläge im Rahmen der Betreuung der Forschungstätigkeit.

Für die hilfreichen Erläuterungen und Diskussionen zu den Anforderungen eines Prüfungsmanagementsystems und die Bereitstellung einer realen Datenbasis möchte ich meinen Kollegen im Institut für Informationssysteme und Softwaretechnik (IFIS, http://www.ifis.uni-passau.de/), insbesondere Dr. Ulrich Zukowski, danken.

Literatur

[GL88] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In Robert A. Kowalski and Kenneth Bowen, editors, *Proceedings of the Fifth International Conference on Logic Programming*, pages 1070–1080, Cambridge, Massachusetts, 1988. The MIT Press.

[Lif] Vladimir Lifschitz. Introduction to answer set programming, unpublished draft.

[Nie99] Ilkka Niemelä. Logic programming with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence*, 25(3,4):241–273, 1999.

[NS00] Ilkka Niemelä and Patrik Simons. Extending the Smodels system with cardinality and weight constraints. In Jack Minker, editor, *Logic-Based Artificial Intelligence*, chapter 21, pages 491–521. Kluwer Academic Publishers, 2000.

[SN01] Tommi Syrjänen and Ilkka Niemelä. The smodels system. In *Proceedings of the 6th International Conference on Logic Programming and Nonmonotonic Reasoning*, Vienna, Austria, September 2001. Springer-Verlag.

[Syr03] Tommi Syrjänen. Logic programming with cardinality constraints. Research Report A86, Helsinki University of Technology, Laboratory for Theoretical Computer Science, Espoo, Finland, December 2003.