Entwicklung eines Tutorensystems für die Logikprogrammierung auf der Grundlage von gewichteten Constraints

Ein Tutorensystem ist ein Computersystem, das dem Student eine Reihe von Übungsaufgaben in einer bestimmten Domäne zur Verfügung stellt und ihn beim Lösen dieser Aufgaben begleitet. Ein solches System gibt dem Student Hinweise für mögliche Verbesserungen an seinen Lösungsversuchen. Durch die Hinweise soll der Student aus seinen Fehlern lernen, und durch das Umsetzen der Hinweise kann der Student seine Fertigkeiten im Problemlösen verbessern. Um dem Student Korrekturhinweise geben zu können, muss das Tutorensystem die Schülerlösung analysieren und mögliche Unzulänglichkeiten erkennen können. Dafür muss das System mit entsprechendem Wissen ausgestattet werden.

Es existieren zwei alternative Ansätze zur Modellierung des Wissens für Tutorensysteme: Model-tracing- und constraint-basierte Techniken. Der Model-tracing-Ansatz benutzt Produktionsregeln, um alternative Lösungspfade und häufige fehlerhafte Pfade für ein gegebenes Problem zu modellieren. Damit stellt der Model-tracing-Ansatz den Prozess des Problemlösens in den Mittelpunkt. Demgegenüber modelliert der constraint-basierte Ansatz die Prinzipien einer Domäne und geforderten Eigenschaften einer korrekten Lösung. Die dafür verwendeten Constraints greifen oftmals auf eine ideale Lösung zurück, um die Richtigkeit einer Studentenlösung zu überprüfen. Auf der Basis dieser beiden Ansätze wurden in zahlreichen Domänen Tutorensystemen entwickelt, unter anderem für LISP, Algebra und SQL.

Programmierprobleme stellen eine große Herausforderung für die Entwicklung eines Tutorensystems dar, weil wir hier mit neuen Anforderungen konfrontiert sind, die in den existierenden Systeme nicht behandelt worden sind. Das sind: 1) die Existenz von alternativen Lösungsstrategien, 2) Die Möglichkeit, eine Lösungsstrategie durch die Anwendung von unterschiedlichen Programmierkonstrukte auf verschiedene Weise zu implementieren, und 3) die Möglichkeit, den Programmtext zu modularisieren. Durch diese Anforderungen wird der Raum der möglichen Lösungen für ein Programmierproblem sehr groß.

Im Prinzip kann auch für Aufgaben aus dem Bereich der Programmierung der Model-tracing-Ansatz verwendet werden, um das Wissen für ein Tutorensystem zu modellieren. Da der Raum der alternativen Lösungspfade für ein Programmierproblem aber sehr groß und der Raum der möglichen fehlerhaften Pfade praktisch unbegrenzt ist, wird der Aufwand für die Modellierungsarbeit enorm.

Unter diesen Bedingungen ist der constraint-basierte Ansatz effektiver, weil nur eine überschaubare Menge von Prinzipien der Logikprogrammierung und von erforderlichen Eigenschaften korrekter Lösungen modelliert werden muss. Allerdings hat dieser Ansatz einige Einschränkungen. Erstens wird in komplexen Domänen häufig eine ideale Lösung gebraucht, die sich in einem großen Lösungsraum mit mehreren alternativen Lösungsstrategien und zahlreichen semantisch äquivalenten Formulierungsvarianten nur schwer identifizieren lässt. Zum zweiten können Korrekturhinweise, die sich auf eine andere als die vom Student beabsichtigte Lösungsstrategie beziehen, zu erheblicher Verwirrung beim Schüler führen. Drittens ist es wünschenswert, Unzulänglichkeiten einer Lösung nach ihrer Wichtigkeit zu priorisieren, um zu entscheiden, welche Korrekturhinweise zuerst angezeigt werden sollen.

Meine Hypothese ist, dass die drei obengenannten Einschränkungen des constraint-basierten Ansatzes durch die Erweiterung um eine heuristische Komponente, die als *Constraint-Gewicht* bezeichnet wird, überwunden werden können. Ein Constraint-Gewicht beschreibt die Wichtigkeit eines Constraints und dient dazu 1) die vom Schüler implementierte Lösungsstrategie zu hypothetisieren, 2) die Suche nach der plausibelsten Fehlererklärung zu steuern, 3) die Fehlererklärungen zu gewichten, falls es für einen Fehler eine Vielzahl von möglichen Erklärungen gibt, und 4) die Korrekturhinweise zu priorisieren. Um die Abhängigkeit von der Verwendung einer idealen Lösung zu verringern, werden darüber hinaus die semantischen Anforderungen für jede der möglichen Lösungsstrategien in relationaler Form modelliert. Diese relationale Darstellungsform für die semantischen Anforderungen ist flexibler als eine ideale Lösung, da sie es erlaubt, die Korrektheitsforderungen für verschiedene Serialisierungsvarianten eines Programs gemeinsam zu erfassen.

Auf der Grundlage von gewichteten Constraints wurde ein Tutorensystem für die Logikprogrammierung entwickelt. Im Rahmen einer offline-Evaluation konnten von insgesamt 221 Schülerlösungen aus früheren Klausuren in 87.9% die implementierte Lösungsstrategie korrekt bestimmt werden. In 92.7% der Fälle, in denen die Lösungsstrategie korrekt identifiziert werden konnte, wurden auch die Fehler korrekt diagnostiziert. Eine Online-Evaluation mit 35 Studenten einer Lehrveranstaltung zur Logikprogrammierung ergab, dass durch die Arbeit mit dem System eine Verbesserung der Fertigkeiten in der Logikprogrammierung mit einer Effektstärke von 0.39 Standardabweichungen erreicht werden konnte.