ＩＳ³における知識獲得手法

和田　正寛　　森下　太朗

シャープ（株）　情報システム研究所

１．はじめに

エキスパートシステム構築において常に問題となるのが、知識ベースの作成である。現在はＫＥの力に頼りきりであるが、人工知能研究の観点から、システムに自ら知識獲得を行わせることが期待されている。既に様々な知識獲得システムあるいは学習システムが発表されてきているが、ここでは我々の実験システム・知的スケジューリングシステムＩＳ³（Intelligent Scheduling System of SHARP）において用いた知識獲得手法を紹介する。

２．ＩＳＳＳ

(1) システム概要

対象問題としては、日時が指定された予定（reserve）・日が指定された予定（day_reserve）・指定のない予定（free）を各参加者のスケジュール表に埋めていくという作業を行う。システムは、自由度のある予定は、空いている場所に適当に入れ、入れられなくなった場合はfailしてbacktrackを行い、別解を捜す。
システムはPrologの拡張言語であるＣＩＬ言語で記述している。

(2) スケジューリング知識

現在対象としているスケジューリング知識としては、置かれた予定（アイテムと呼んでいる。）の取る状態の禁止知識（制約知識）である。すなわち、「アイテムＡは時刻Ｔになってはならない。」「アイテムＡとアイテムＢは連続してはならない。」といったような知識を扱っている。

(3) 知識獲得

ユーザーは、システムが提示してきたスケジュール表に対して、不満なアイテムの場所をマウスクリックする（図１）。システムは、内部に『制約 template 』と呼ぶ、知識の骨組みを持っている（図２）。システムは、ユーザーの指示したアイテムの状態が、これらの template に当てはまるかどうかを調べる。そして、当てはまる template が見付かったならば、template の不定項目をそのアイテムの状態で充足し、知識として完成させる。
ただし、このようにして作られた知識はまだ仮説に過ぎない。仮説が見付かると、システムはその仮説を作成したスケジュール表全体に適用してみる。そして、その仮説に反する結果が表の他の場所に存在しないかどうかを調べる。もしも反例が存在したならば、ユーザーが今その仮説を意図している可能性は低いと判断し、その仮説のプライオリティを下げる。
この様にして優先順序が付けられた仮説を、システムはその順序にしたがって、意図しているものであるかどうかをユーザーに問いかけてくる（図３）。優先順序を付けることによって問いかけの回数を減らすことが期待される。

(4) 一般化・特殊化

template 検索を行う時、アイテムそのもので検索を行うと、「Ｋ氏との面会は９時になってはならない。」といった仮説が生成され、反例の極めて少ない特殊な仮説が多く出来る。そのため反例チェックの長所が生かされなくなる。また、そのような知識はかなり特殊な、融通性のない知識となってしまい、知識ベース維持の面でも問題が多い。
そこでこのシステムでは、template 検索を行う際は必ず、アイテムの一段上の概念クラスで検索を行う戦略を取っている。すなわち、『Ｋ氏との面会』であれば『面会』、『Ｘについての会議』であれば『会議』といった概念クラスで検索を行う（図４）。そしてアイテムそのものを適用した仮説は、それらの一般的な仮説よりも優先順序を低くしておく。そうすると、一般的な仮説の問い合わせが失敗した時、それらを特殊化した仮説として問い合わせる動作を実現出来る。
また、仮説がユーザーによって肯定された場合、システムはその知識の一般化を試みる。すなわち、現在成立している知識を、さらに上の概念クラスにおいて仮説として生成する。そして初めの仮説が生成された時と同様、スケジュール表全体に適用してみて反例チェックを行う。

(5) ステップトレース

この知識獲得手法では、「ユーザーの指摘したところに特異的に存在している状態が、ユーザーの意図である。」という考えを中心的な戦略としている。そのため、獲得しようとしている知識に反例が存在した場合、優先順序が低くなってなかなか仮説が出てこない。よってそのような事態を避けるため、各アイテムが置かれていくのをステップトレースし、問題となる状態が発生した時点で指摘する方法を取れば、効果的な動作が得られる。

A Knowledge Acquisition method on IS³

Masahiro HADA (mada@islis.sharp.junet) , Taro MORISHITA

SHARP Corporation Information System Laboratories

(ii) 制約 template

この知識獲得手法で最大の問題となるのが、制約 template の存在である。即ち、「誰が、いつ、制約 template を記備するのか。」という問題が残っており、はじめに述べた「知識ベース構築の問題」は根本的には解決されていない。しかし、次の二つの点で、この手法に意義があると考えている。

[Ⅰ] 制約 template は、文字通り知識の骨組みである。専門家あるいはKEは、そのシステムが取り扱う問題において有効と思われる知識の枠組みだけを記述する。実際にその枠組みをどのように使用するかは、ユーザーにまかされる。よって基本知識（制約 template）は広汎に用いられ、また実システムを構築する時、システムは知識のカスタマイズの支援を行っている。

[Ⅱ] 制約 template は、極めて単純な形をしている。すなわち、問題を表現している要素における属性間の関係が基本で、その関係も数値の大小や、member の存在や、list の長さ等を 調べる評価関数からなっている。よって、問題を表現する要素の構造から、template 作成もシステムが支援出来るのではないかと期待している。

3. ISSSの課題

現在のシステムの最大の弱点は、「～という状態にする。」といった知識（戦略知識）を扱えない点である。つまり、アイテムを配置する時にまったく知識を使わずに場所を仮定し、極めて単純な generate & test しか行っていない事である。実際のスケジューリング作業では、そのような非効率な事はしていない。『アイテムの配置場所の優先順序を決められるようにする事は簡単だが、その順序概念をどのようにシステムに認識させるかが難しい。』

また、現状ではスケジューリングが不可能になった時、システムはユーザーに修正をまかせるほかないが、制約を扱っている以上、『制約緩和の枠組みも持たせる必要がある』。そのようなメタ的な知識をいかに獲得させるかを考慮しなければならない。

4. むすび

この研究は財団法人 新世代コンピュータ技術開発機構の研究テーマの一環として行われた。

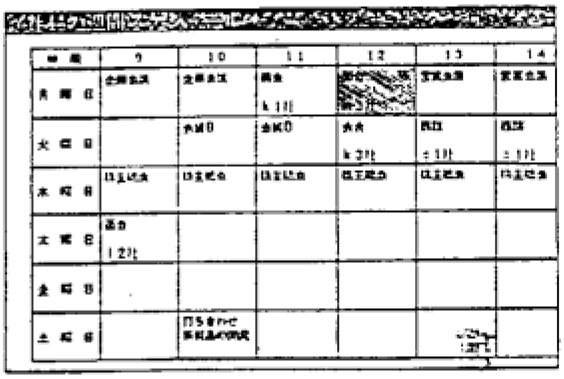

図1：システムが作成したスケジュール表（部分）

※ 単一アイテム制約

※ Rが参加するXがD曜日になるのを禁止する。
[item] X, [day] D, [chosen_member] R.

※ Rが参加するXが午前になるのを禁止する。
[item] X, [time] T, [chosen_member] R, gozen (T).

※ 相互アイテム制約

※ X1がX2と同じ日になるのを禁止する。
[item] X, [item] X1, sameday (X, X1).

※ X1がX2の翌日になるのを禁止する。
[item] X, [item] X1, nextday (X, X1).

図2：制約 template ベース（部分）

対話を開始します。

>>［a社長］の参加する［面会］が
　［12時］になるのを禁止しますか？
<<y

>>［a社長］の参加する［一般会合］が
　［12時］になるのを禁止しますか？
<<y

対話を終了します。

図3：システムとユーザーとの対話例

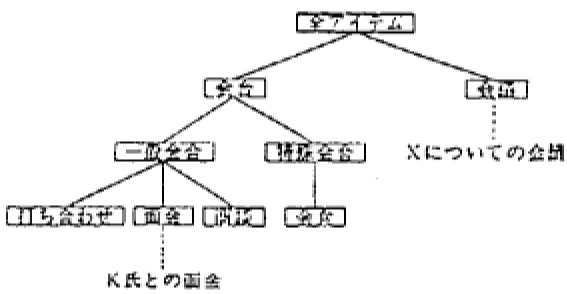

図4：概念クラス階層図