知識工学と人工知能 47－10
（1986. 7. 11）

物語理解システムの試作

市坡 槙子

東京工業大学理学部情報科学専攻

非単調な推論をサポートするシステムTMS [Doyle 79] 上に物語理解システムを試作した。その特徴は、各登場人物の信念空間を用いて物語を計算機上にモデル化し、そのモデルをTMSの信念修正メカニズムの下で管理することである。本システムでは読み手の予測や登場人物の信念の状態を、非単調な推論としてモデルの中に積極的に取り込み、理解過程で利用していくことができる。また各々の信念空間の間にある矛盾等の情報についても、非単調な推論を利用して管理する。

さらに本システムでは、スクリプトの記述法についても工夫がなされている。即ちスクリプト内に分岐を設けて逸脱時等に複数の可能性を扱う際、単なる場合分けのための条件分岐でなく、スクリプト全体の信念の修正を伴う分岐を可能にした。これと関連してスクリプト内の事象の依存関係を記述する方法を新たに考案した。この記法では、逸脱処理のため上位スクリプト等関連知識を呼出す場合も、必要な時にデーモンが事象間の依存関係を張ることで効率的に対処できる。

Construction of Experimental Story Understanding System

Yoshiko ICHIBA

Tokyo Institute of Technology

A story understanding system using non-monotonic reasoning is described. The system controls overall structure of various knowledge obtained from input story and uses TMS, written by Doyle [Doyle 79], to carry out actual non-monotonic reasoning processing.

The system maintains multiple belief spaces for each person in a story, along with one real space ,which stands for "absolute truth" about the story. As each sentence in a story is fed to the system, new knowledges are added to those belief spaces, and integrity of the whole state of the system is maintained through TMS's belief revision mechanism. By making use of belief revision mechanism and separated belief space for each person, the system appropriately handles reader's expectation and belief states of each person in the story. Especially, a new method to deal with contradiction between each person's belief spaces is proposed.

In addition, a more flexible model of representing script knowledge using non-monotonic reasoning is examined. In this method, a script can handle multiple paths and branches at any point. And such branches can cause revision of multiple states wholly over the script and belief spaces. This states differs from the conventional script mechanism, in which simple conditional branching at some small set of discriminated states were possible. To achieve such a flexible mechanism, a new method to represent dependency of events in a script is proposed. In this method, when one knowledge requested invocation of some relative knowledge accompanying it (superordinate knowledge, etc.), a daemon process may be invoked to construct dependency links between related events. With this method, dependency links can be managed efficiently.

研究の背景

人が物語を理解する過程は、読み手が持っている実世界に関する知識や常識、物語の展開についての自分の期待などを総動員して物語文を結合していく作業である。日常的な出来事の多くは、過去の経験や学習を通して既に得られた知識を駆使することで予想の範囲内にあったり、納得されるものである。しかしこの種の知識を計算機上に表現して物語を理解させることを考えると、「レストランで食事する」というような日常的な場面さえ、様々な出来事と結びつく可能性があり、それをすべてうまく表現し、また実際に効率よく推論を進めることは難しい。さらに、事態の我々が常識的と了解しているなりゆきからの逸脱をそれと認識し、その特異性に注目し理解することもできなければならない。

このような問題に取り組むため、物語を理解する過程を計算機上でシミュレーションすることを試みたが、その際図１のような（物語理解のための）知識利用モデルを用いた。

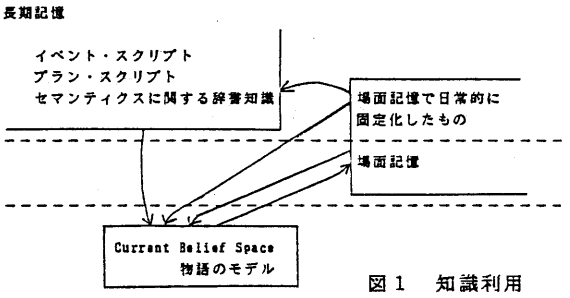


図１　知識利用モデル

長期記憶は読み手の持つ一般的な知識や常識が蓄えられている場所であり、長期記憶の知識を用いて物語文を組み立てる場が Current Belief Space である。本システムでは長期記憶に相当する知識は、階層化されたスクリプトと語の一般的な意味に関する辞書の知識からなっているので、アプローチとしてはいわゆるスキーマ・ベースの理解モデルに基づいている。さらに長期記憶と Current Belief Space の中間に、場面記憶のレベルを考える。場面記憶は、「〇月〇日何処々々で何々があった」というような、特定の出来事を記憶する場所である。ここの記憶は、しばらくすると減衰するかもしれないし、「行きつけのお店に関するスクリプト」となって、長期記憶の一部になるかもしれない。物語中でスクリプトが閉じたり、場面が変わったりすると、 Current Belief Space の一部は場面記憶に移るが、その後再び呼び出されるなど、Current Belief Space と場面記憶との間には相互作用がある。同様に繰り返し起こる出来事から、一般的な知識を学びとるなど、長期記憶と場面記憶との間にも相互作用がある。このような、スキーマの出し入れやスキーマ同士の類似性などは、スキーマ・ベースのアプローチにとっては将来の大きな課題であろうが、本システムではその手始めとして、長期記憶にある知識をCurrent Belief Spaceで利用する仕組みと、効率のよい利用を可能にするための知識表現の方法に主眼を置いている。以下ではまず本システムにおいて採用した信念の修正メカニズムの有用性について論じ、続いて試作した物語理解システムについて報告する。

１．信念の修正メカニズムの応用

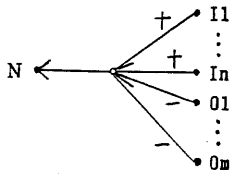
McDermott 等は[McDermott&Doyle 80]信念の修正メカニズムを非単調論理として定式化した。また、Doyle [Doyle 79]は非単調論理をサポートするデータベース・システムＴＭＳ(Truth Maintenance System)を実現した。本研究の物語理解システムはＴＭＳ上に築かれているので、まず次節でＴＭＳについて簡単に説明し、記法についての約束をする。

1-1.ＴＭＳ

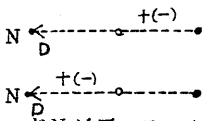
ＴＭＳは推論を行なうプログラムのもとで動作し、推論される知識(beliefs) をノードと呼ばれるデータ構造として記録し、truth maintenance やDependency Directed Backtrackingと呼ばれる管理を行なうデータベースである。ノードはすべて、IN（信じられている）かOUT （信じられていない）のどちらかの状態を持つ。ノードの状態は、そのノードに付加されているjustification が評価されて決まる。justification は記法 (SL (I1,I2,..,In) (O1,O2,..,Om))で表わし、データベース中でノードI1～In(insupport justification)がすべてIN、かつO1～Om(outsupport justification) がすべてOUT の時、そのjustification をもつノードがINになることを意味する。ひとつのノードが複数のjustification を持つ時は、どれかひとつがINであればノードの状態はINである。(SL ()()) 、つまり m=n=0であるようなjustification をもつノードは、premise といい、常にINである。本システムでは物語の入力文そのものはpremise として扱っている。一方(SL (I1,..,In) (非空 O1,..,Om)) 、m>0 であるようなjustification にサポートされているノードをASSUMPTIONというが、これはO1,..,Omの事象が信じられていないという仮設に基づいてそのノードを信じることを意味するので、後で別の仮定が採用されると真偽値が変化する。

ＴＭＳのユーザは、データベース内に新しくノードを作ったり、ノードのjustification を付加／削除する。この操作の結果、あるノードの真偽値が変化した場合、それに伴うデータベース全体の真偽値の修正を行なう機能がtruth maintenance である。また、ユーザは推論プログラムからＴＭＳデータベース中のあるノードの矛盾を告げて、ＴＭＳの Dependency Directed Backtracking 機能を起動することができる。この機能は、矛盾ノードのjustification を逆のぼって、矛盾ノードをサポートするASSUMPTIONタイプのノ

ードを見つけ出し、そのASSUMPTIONのjustification中のOUT ノードのどれかひとつをINにして（つまり別の仮定を採用して）、矛盾の解消を図る。これらの機能に加えて、ＴＭＳ中の各ノードにデーモンを付加し、そのノードの状態がINからOUT 、OUT からINに変化した時に、指定した手続きを起動することができる。以上の機能を備えた、ＴＭＳ中のノードの依存関係を、便宜上次のようなグラフで表わすことにする。



・ノードNが(SL (I1,..,In) (O1,..,Om))
を持つことを示す



・ノードNがデーモンを持ち、それが起動されると点線のような依存関係を張ることを示す

図２　ＴＭＳ内ノードの依存関係を表わす記法

1-2.物語理解システムの概要

本システムはVAX-11/780のFranz Lispの上で作成した。本研究では先に述べたように、物語理解過程で必要となる知識の表現や利用方法に主眼を置いたものであり、物語文の入力形式及びシステでの物語文の内部形式は、次のような格文法理論に基づいた深層構造を表わす文と呼ぶ単位の集まりである（現在は自然言語の文章をそのまま入力するわけではない）。

文　＝　（(ﾓｰﾄﾞ +/not)　(PRED 動詞)
　　(AGENT　主格)　（格名　値）＊）
ただし、＊は任意回の繰り返しの意

システムは１文に対応する事実をＴＭＳの１ノードに割り付ける。入力文はまず、客観的事実を保持する「現実空間」にINノードとして投入され、さらに物語の状況に応じて、適切な「登場人物の信念空間」に投入される（図３入力部）。次に入力文に対し辞書引きによる意味処理を行なう。その時必要ならば、入力文に基づいてスクリプトを起動し、スクリプトに書かれている知識を信念空間に投入し、入力文を適切な文脈の中で理解しながら、物語のモデルを形成する（図３理解部）。語彙のセマンティクスに基づく推論やスクリプトの呼び出しに関する方法は、シャンク等の方法に従っている。物語を理解した結果は、入力文ごとの、信念空間内のノード（各事実に対応）の状態遷移の系列として表現される。

1-3.物語の計算機内モデル

出力される読み手（計算機）の物語のモデルは、物語の述べる事実を取り込む現実空間と各登場人物の信念の状態を示す信念空間を、図４、５の例のようにＴＭＳ上に形成したものである。例はそれぞれ文章「デラはジムにプレゼントを送りたい」「デラは髪を切る」に対応しており、２節の例題の１文である。

現実空間は入力文等、実際に起こった事柄を記録するだけでなく、その各々の事実を誰が信じていて誰が信じていないかを示す情報を管理する役割をもつ（図５のようにすれば、後でジムが事実を知ったことも現実空間によって管理される）。このようにしたのは、各登場人物それぞれの信念の状態も重要であるが、物語全体としては登場人物同士の信念状態の食い違いなどが重要な意味を持つことがあるからである。

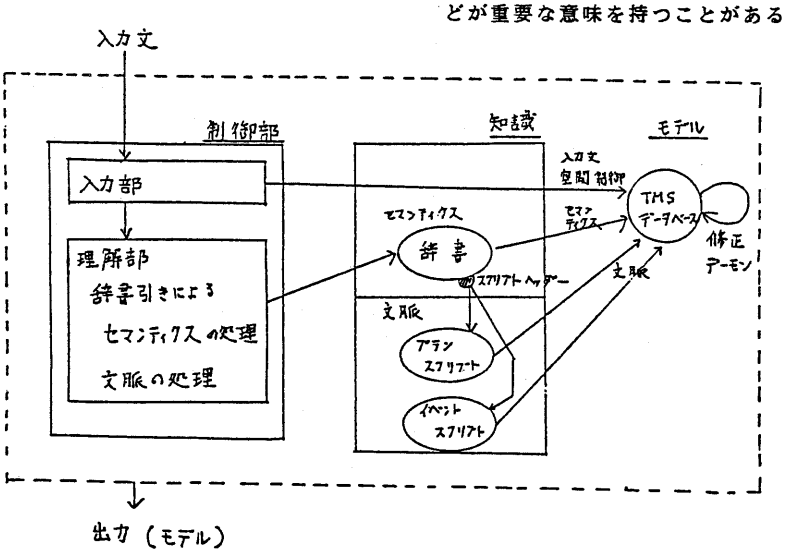


図３
物語理解システム
の概要

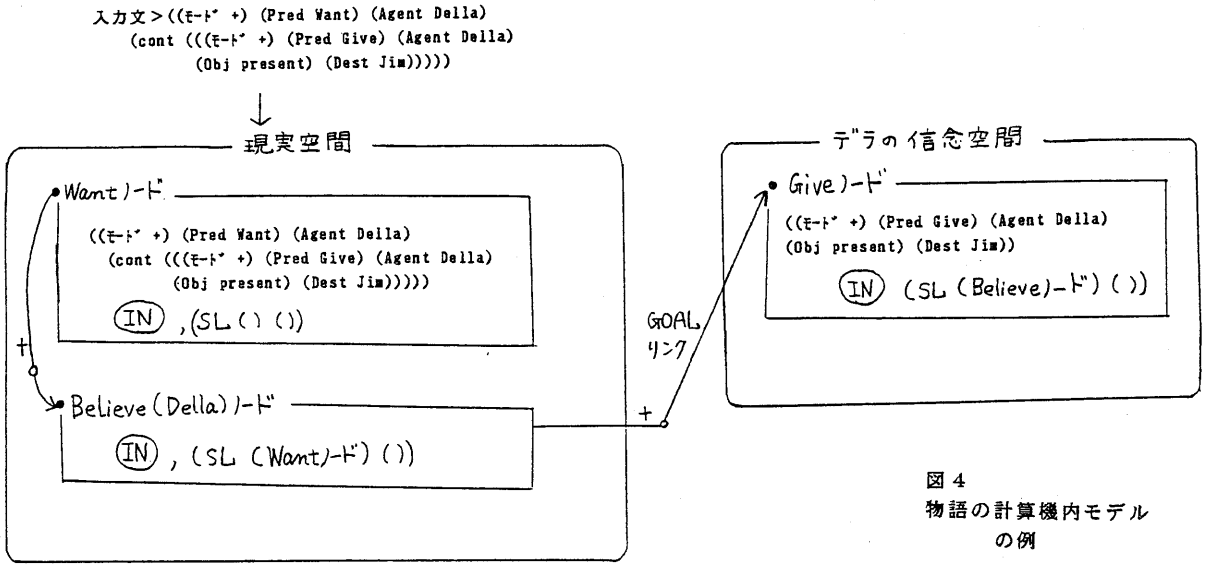


図４
物語の計算機内モデル
の例

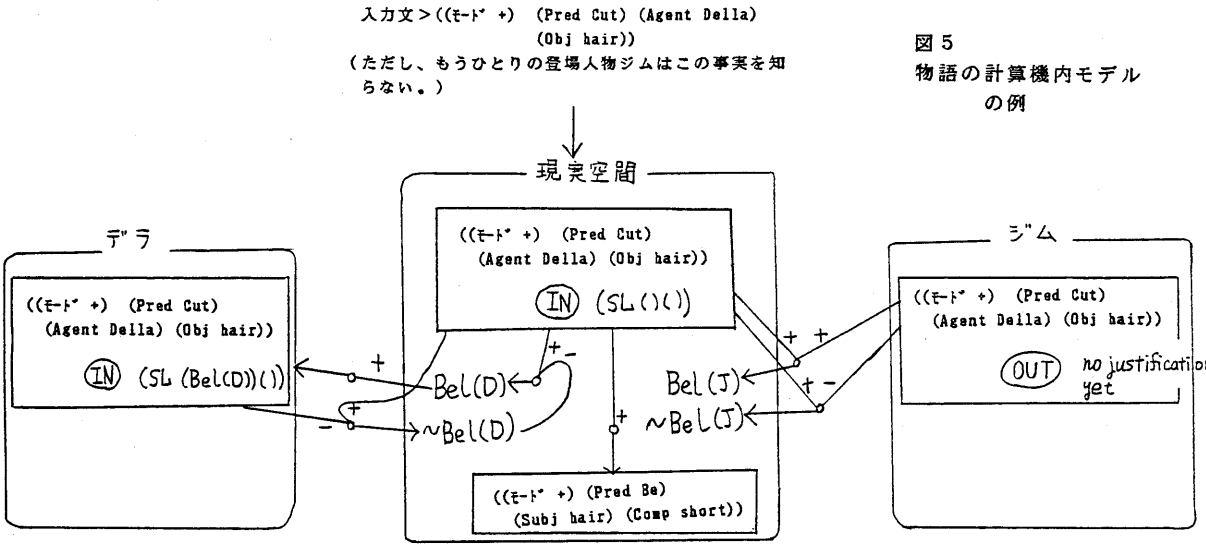


図５
物語の計算機内モデル
の例

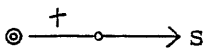
1-4. ７つの推論パターン

スクリプトや辞書中の語彙に関する知識に伴う推論は、次の７つの推論パターンにより記述した。各パターンの記法とＴＭＳ上での実現方法を説明する。ただし、以下で◎はその推論が起動された時に処理されている文（事実）、Ｓは１文、Ｓset ={S1,..,Sn} は文の集まりを示す。また、実現上必要なCREATE述語は次の機能で、新情報を旧情報に結合させ、各空間を形成する。

［用法］(CREATE space 文)

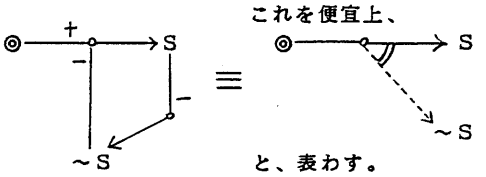
［機能］指定された空間内に、文の示す事象が既に存在するかマッチングを行ない、あればそのノードをかえし、なければ文の示す事象に相当するノードを新しく生成して（ただし状態はOUT ）返す。マッチングの時、２つの事象が互いに一般化や具体化の関係にあるかどうかも判定を行ない処理するが、本稿では詳細を述べない。

(1) ◎ －＞S



Ｓノードのinsupport justification に◎ノード。

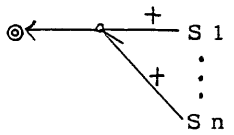
(2) ◎＊－＞S



常に、Sか～Sかどちらか一方のみINになるよう管理される。（[Doyle 79]のASSUME機能）。

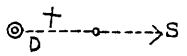
(3) ◎＜－Sset

create S1, create S2, .. , create Snを行なった後で、◎ノードのjustification を以下のように付加する。

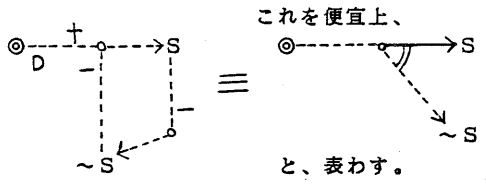


(4) ◎　－>>S

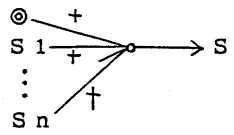
◎ノードのjustification をデーモンを用いて以下のように定める。



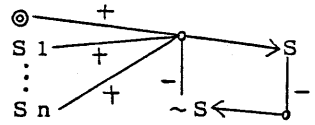
(5) ◎＊－>>S



(6) ◎, Sset　－＞S



(7) ◎, Sset ＊－＞S



(3) は後ろ向き推論で、それ以外は前向き推論であるが、＊付き（これをASSUME機能という）の推論は「～かもしれない」という推論を表わし、後で覆る可能性がある。デーモンにより実行される推論は、別の文脈への移行やある事象に焦点があたった場合についてのトリガーを用意するのに有効であり、膨大な常識等の知識を効率良く表現・利用するのに重要である。

(1) ～(7) の推論の各リンクには、推論の種類を表わすラベルが付けられ、そのリンクをたどって推論が行なわれる。ラベルには、因果関係をあらわすREASONリンク、時間的順序関係を示すPRECEDE リンク等がある。

現在、上の記法からＴＭＳオペレーションへの変換プログラムを作成し、スクリプトや辞書項目を記述するツールとして用いている。

1-5.スクリプトの記述例

上述の７つの推論パターンを用いて、簡単化したレストラン・スクリプトを記述すると図６のようになる。（レストラン・スクリプトが起動されている時の「食べ物の持ち帰り」のような状態変化のトリガーとなる事象の実現と、perspective の切り換えの実現が例示されている。）基本的にスクリプト内のすべて事象はASSUME機能で生成され場合分けされいて、いつでも逸脱に対する準備がある。特に重要と思われる事象にはデーモンがいて、そのノードが新たに信じられたり信じられなくなったりした時に、その説明を与えるような他の知識を呼び出したりして推論をすすめる。

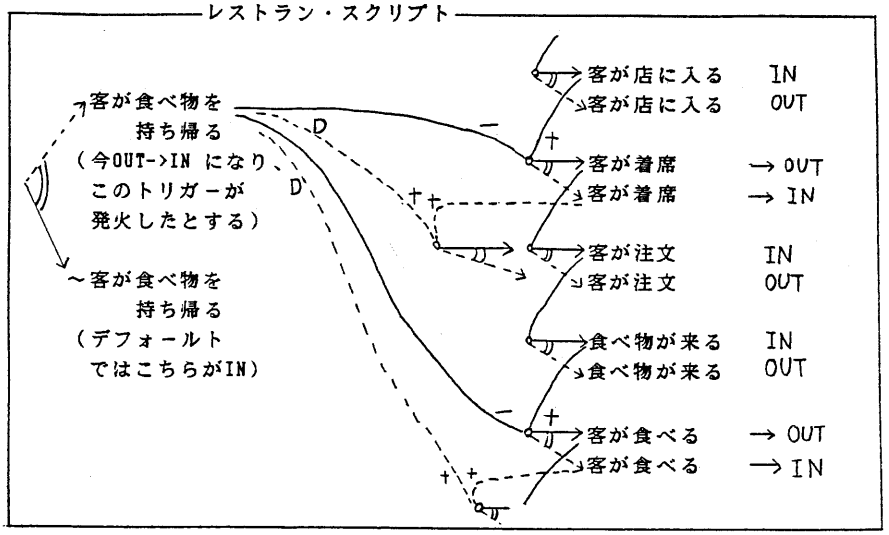
ＢＯＲＩＳシステム[Dyer 83] では、スクリプト内の事象に上位概念への糸をつけることで、あるひとつのスクリプト内の事象でも必要ならばそのスクリプトを超えて別の知識と相互作用をもつことができる。そしてこれによって、perspective の切り換えや逸脱の処理を行なう。本稿の方法は、ASSUME機能を用いることでこの考えをさらに徹底したものである。スクリプト内の事象といえども各々いつ逸脱してもよく、トリガーが用意されていて逸脱の理由に心当たりのある時は、デーモンがボトム・アップにその知識を呼び出すようになっている。また、その場の意外な突発事故など心当たりがなくて逸脱の説明がつかない場合でも、スクリプト中の当然起きるべき事象が起こらなければ、それに伴って起こらなかったはずのことは、PRECEDE リンク等の記述により自動的に予測からはずれる。

2．物語理解システムの動作例

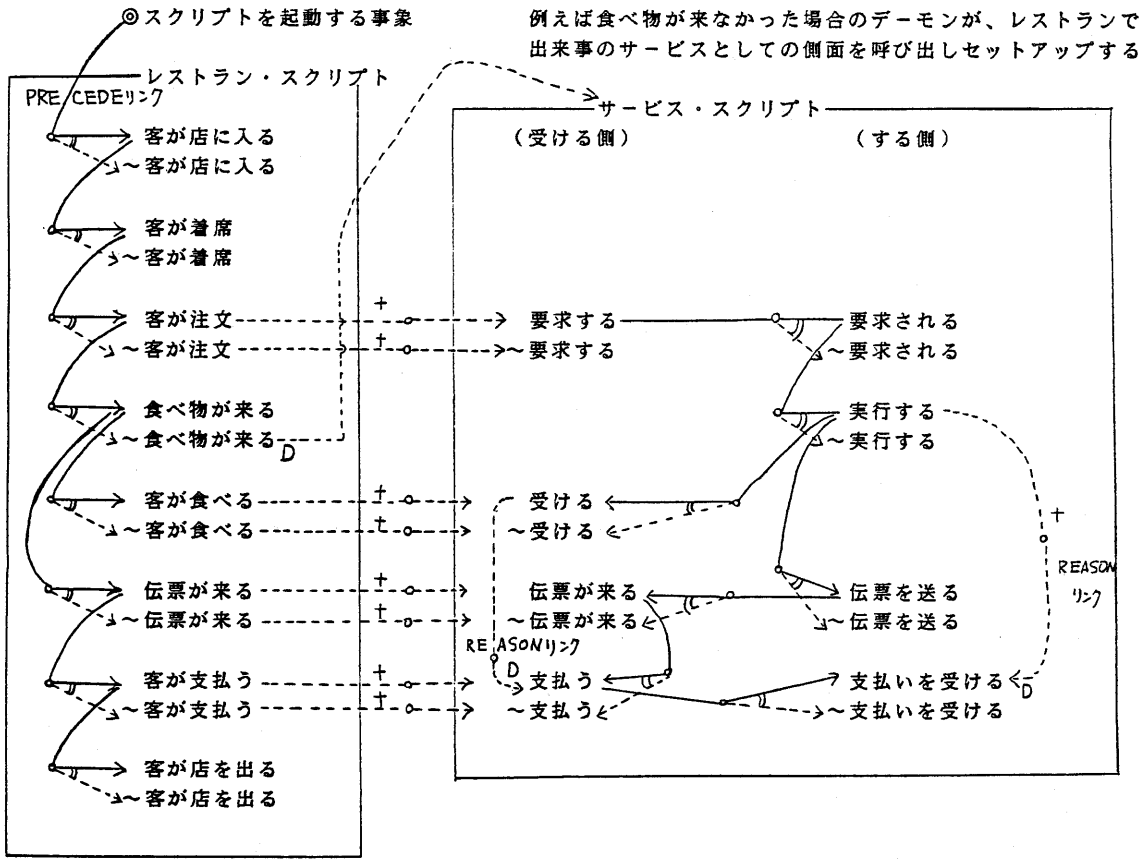
2-1.入力例

実験は、次の例題を上で述べた内部表現の形式に変換したものを入力した。

例題　：ジムはデラの夫である。クリスマスの日のことである。デラはジムになにか贈り物がしたいと思った。しかし、デラはお金がなくて、贈り物を買うことができない。ところで、デラは長くてとても美しい髪をしていて、それを大切にしていた。デラは悲しかったけれども、その髪を切り、売って、お金にかえた。そして時計屋で時計のベルトを買った。さて、ジムが帰って来た。驚いた様子のジムにデラは、”髪を売っ



(a)トリガー事象の発火によるスクリプト修正の実現の例



(b) perspective の実現の例

図6 レストラン・スクリプトの記述例

注．特にデーモンのいない事象はASSUMEの代わりに
CREATEを使えば、否定のノードは生成されない。

て、このプレゼントを買ったのよ”と言って、ジムに贈り物を渡した。ジムはとても喜んだ。

入力文：

```
(input '(((mode +)
         (Pred Be)
         (Agent Jim)
         (Complement husband)
         (Of Della))          --- 文1
        0))

(input '(((mode +) (Pred WillBe) (Agent ***) (Complement Christmas)) 0))
(input '(((mode +) (Pred Want) (Agent Della0)
                   (cont ((((mode +) (Pred Give) (Agent Della0)
                                     (Obj present) (Dest Jim0)) 0)))) 0)))
(input '(Because ((((mode not) (Pred Have) (Agent Della0) (Obj money))
                   (PreBecause)))
                 ((((mode cannot) (Pred Buy) (Agent Della0) (Obj present0))
                   (ConBecause)))))

(input '(((mode +) (Pred Have) (Agent Della0) (Obj hair)) 0))
(input '(((mode +) (Pred Be) (Agent hair0) (Complement beautiful)) 0))
(input '(((mode +) (Pred Be) (Agent hair0) (Complement long)) 0))

(input '(((mode +) (Pred Cut) (Agent Della0) (Obj hair0)) 0))
(input '(((mode +) (Pred Sell) (Agent Della0) (Obj hair0) (Dest ***)) 0))
(input '(((mode +) (Pred Get) (Agent Della0) (Obj money)) 0))
(input '(((mode +) (Pred Buy) (Agent Della0)
                   (Obj watchband) (Source WatchStore)) 0))

(input '(((mode +) (Pred Come) (Agent Jim0) (Dest home)) 0))
(input '(((mode +) (Pred Say) (Agent Della0) (Dest Jim0)
                   (cont ((((mode +) (Pred Sell) (Agent Della0)
                                     (Obj hair0) (Dest ***)) 0)
                          (((mode +) (Pred Buy) (Agent Della0)
                                     (Obj present0) (Source ***)) 0))))
        0))
(input '(((mode +) (Pred Give) (Agent Della0) (Obj present0) (Dest Jim0))
(input '(((mode +) (Pred Be) (Agent Jim0) (Complement happy)) 0))
```

図７　入力例

注　hair0 や present0 は、一般的な髪やプレゼントではなく、ある特定の個体を指し示す。

2-2.出力例

出力は、各空間に生成されたすべてのノードとその依存関係を示す情報である。図８に例題に対する出力の冒頭の部分を示す。これは文１を"husband"の語彙のセマンティクスが、結婚に関する知識を呼び出し推論を行なって処理したものである。ただし、

exn:ノードの名前

st :ノードの状態

justifications: ノードのjustification
((I1,...,In) (O1,...,Om))

jus-argument　: ノードに割りつけられた事象の内容（文）

を意味する。

2-3.考察

本例題のために記述したスクリプトは、

物を売る場面のスクリプト（イベントスクリプト）

物を買う　　〃

人にプレゼントをしたい時どうするかを記述したスクリプト（プランスクリプト）

物を買いたい時　　〃

お金が必要な時　　〃

である。スクリプトの起動はSAM 同様、スクリプト・ヘッダーを満たしているかどうかをチェックして行なう。

スクリプトや語彙のセマンティクスによる推論の結果が、既に築かれた物語のモデル上の事象とマッチして取り込まれたり、また逆に将来の入力文に対する予測や説明となりながら、物語文を結合していく。その際、読み手が持っている知識や予測の状態を、後の入力に応じて望ましいように切り換えられるような仕組みが必要である。この点で、スクリプトの知識をはじめ物語モデルを形成する推論の過程を、信念の修正機能を備えたＴＭＳ上に実現する方法は有効であった。

３．結論

スクリプトのような一群のまとまった知識は、その中の各事象が時間的、因果的に複雑に依存し合っているので、その事象間の関連をうまく表現してやらなければならない。本文ではスクリプト内の条件分岐や、そのスクリプトからの逸脱を、各事象に与えるjustification setによって、スクリプト全体の信念の修正を伴う分岐を行なわせるという方法を提案した。このようなスクリプト内の事象の「依存関係」を記述する方法によって、入力文がスクリプト内の事象とマッチングさえされれば、直ちにその入力文によって意図されるスクリプトの利用状態を得、入力文をスクリプトの中で位置付けることができる。

以上の信念の修正を用いるスクリプト表現に関すると同様のことが、Current Belief Space上で、物語文を１文づつ採り込んで、物語のモデルを形成していく過程でも有効である。

謝辞

日頃御指導頂いている木村　泉教授と久野　靖助手に深く感謝致します。また、貴重な資料を紹介して下さった米澤明憲助教授にもお礼を申し上げます。

参考文献

[Doyle 79]　Doyle,J.,A truth maintenance system, Artificial Intelligence12,(3),1979

[McDermott&Doyle 80]　McDermott,D and Doyle,J., Non-monotonic logic I, Artificial Intelligence 13,(1),1980

[Dyer 83]　Dyer,M,G.,IN-DEPTH-UNDERSTANDING., MIT Press,1983

図8　出力例


DellaSpace
1
exn:Be2
st:in
argument:(((mode +)
(Pred Be) (Agent Jim0) (Complement husband0) (Of Della0)) nil)
justifications:(((+Believe0))
jus-argument:(((mode +) (Pred Be) (Agent Jim0)
(Complement husband0) (Of Della0)) nil)
justifications:((MarriedWith2))
jus-argument:(((mode +) (Pred Be) (Agent Jim0) (Complement husband)
(Of Della0)) (EQUIV))
2
exn:MarriedWith2
st:in
argument:(((mode +)
(Pred MarriedWith) (Agent Jim0) (Accompany Della0)) nil)
justifications:((+Believe1))
jus-argument:(((mode +) (Pred MarriedWith)
(Agent Jim0) (Accompany Della0)) nil)
justifications:((Be2))
jus-argument:(((mode +) (Pred MarriedWith)
(Agent Jim0) (Accompany Della0)) nil)
3
exn:MarriedWith3
st:in
argument:(((mode +)
(Pred MarriedWith) (Agent Della0) (Accompany Jim0)) (EQUIV))
justifications:((+Believe2))
jus-argument:(((mode +) (Pred MarriedWith) (Agent Della0)
(Accompany Jim0)) (EQUIV))
justifications:((MarriedWith2))
jus-argument:(((mode +) (Pred MarriedWith) (Agent Della0)
(Accompany Jim0)) (EQUIV))
4
exn:Be3
st:in
argument:(((mode +)
(Pred Be) (Agent Della0) (Complememt wife) (Of Jim0)) (EQUIV))
justifications:((+Believe3))
jus-argument:(((mode +) (Pred Be) (Agent Della0)
(Complememt wife) (Of Jim0)) (EQUIV))
justifications:((MarriedWith2))
jus-argument:(((mode +) (Pred Be) (Agent Della0)
(Complememt wife) (Of Jim0)) (EQUIV))
5
exn:Love0
st:in
argument:(((mode +) (Pred Love) (Agent Jim0) (Obj Della0)) (REASON))
justifications:((MarriedWith2) Love1)
jus-argument:(((mode +) (Pred Love)
(Agent Jim0) (Obj Della0)) (REASON))
6
exn:Love1
st:out
argument:(((mode not)
(Pred Love) (Agent Jim0) (Obj Della0)) (anti-ASSUMP))
justifications:(nil Love0)
jus-argument:(((mode not) (Pred Love) (Agent Jim0)
(Obj Della0)) (anti-ASSUMP))
JimSpace
1
exn:Be4
st:in
argument:(((mode +)
(Pred Be) (Agent Jim0) (Complement husband0) (Of Della0)) nil)
justifications:(((+Believe4))
jus-argument:(((mode +) (Pred Be) (Agent Jim0)
(Complement husband0) (Of Della0)) nil)
justifications:((MarriedWith4))
jus-argument:(((mode +) (Pred Be) (Agent Jim0) (Complement husband)
(Of Della0)) (EQUIV))
2
exn:MarriedWith4
st:in
argument:(((mode +)
(Pred MarriedWith) (Agent Jim0) (Accompany Della0)) nil)
justifications:(((+Believe5))
jus-argument:(((mode +) (Pred MarriedWith)
(Agent Jim0) (Accompany Della0)) nil)
justifications:((Be4))
jus-argument:(((mode +) (Pred MarriedWith)
(Agent Jim0) (Accompany Della0)) nil)
3
exn:MarriedWith5
st:in
argument:(((mode +)
(Pred MarriedWith) (Agent Della0) (Accompany Jim0)) (EQUIV))
justifications:((+Believe6))
jus-argument:(((mode +) (Pred MarriedWith) (Agent Della0)
(Accompany Jim0)) (EQUIV))
justifications:((MarriedWith4))
jus-argument:(((mode +) (Pred MarriedWith) (Agent Della0)
(Accompany Jim0)) (EQUIV))
4
exn:Be5
st:in
argument:(((mode +)
(Pred Be) (Agent Della0) (Complememt wife) (Of Jim0)) (EQUIV))
justifications:((+Believe7))
jus-argument:(((mode +) (Pred Be) (Agent Della0)
(Complememt wife) (Of Jim0)) (EQUIV))
justifications:((Marrie lWith4))
jus-argument:(((mode +) (Pred Be) (Agent Della0)
(Complememt wife) (Of Jim0)) (EQUIV))
5
exn:Love4
st:in
argument:(((mode +) (Pred Love) (Agent Jim0) (Obj Della0)) (REASON))
justifications:((MarriedWith4) Love5)
jus-argument:(((mode +) (Pred Love)
(Agent Jim0) (Obj Della0)) (REASON))
6
exn:Love5
st:out
argument:(((mode not)
(Pred Love) (Agent Jim0) (Obj Della0)) (anti-ASSUMP))
justifications:(nil Love4)
jus-argument:(((mode not) (Pred Love) (Agent Jim0)
(Obj Della0)) (anti-ASSUMP))
RealSpace
1
exn:Be0
st:in
argument:(((mode +)
(Pred Be) (Agent Jim0) (Complement husband0) (Of Della0)) (input))
justifications:(nil)
jus-argument:(((mode +) (Pred Be) (Agent Jim0) (Complement husband0)
(Of Della0)) (input))
justifications:((MarriedWith0))
jus-argument:(((mode +) (Pred Be) (Agent Jim0) (Complement husband)
(Of Della0)) (EQUIV))
2
exn:MarriedWith0
st:in
argument:(((mode +)
(Pred MarriedWith) (Agent Jim0) (Accompany Della0)) nil)
justifications:((Be0))
jus-argument:(((mode +) (Pred MarriedWith)
(Agent Jim0) (Accompany Della0)) nil)
3
exn:MarriedWith1
st:in
argument:(((mode +)
(Pred MarriedWith) (Agent Della0) (Accompany Jim0)) (EQUIV))
justifications:((MarriedWith0))
jus-argument:(((mode +) (Pred MarriedWith) (Agent Della0)
(Accompany Jim0)) (EQUIV))
4
exn:Be1
st:in
argument:(((mode +)
(Pred Be) (Agent Della0) (Complememt wife) (Of Jim0)) (EQUIV))
justifications:((MarriedWith0))
jus-argument:(((mode +) (Pred Be) (Agent Della0)
(Complememt wife) (Of Jim0)) (EQUIV))
5
exn:+Believe0
st:in
argument:(((Pred +Believe) (Agent DellaSpace)) nil)
justifications:((Be0) -Believe0)
jus-argument:(((Pred +Believe) (Agent DellaSpace) (((mode +) (Pred Be) (
Agent Jim0) (Complement husband0) (Of Della0)) nil)))
6
exn:-Believe0
st:out
argument:(((Pred -Believe) (Agent DellaSpace)) nil)
justifications:((Be0) Be2)
jus-argument:(((Pred -Believe) (Agent DellaSpace) (((mode +) (Pred Be) (
Agent Jim0) (Complement husband0) (Of Della0)) nil)))
7
exn:+Believe1
st:in
argument:(((Pred +Believe) (Agent DellaSpace)) nil)
justifications:((MarriedWith0) -Believe1)
jus-argument:(((Pred +Believe) (Agent DellaSpace) (((mode +) (Pred Marri
edWith) (Agent Jim0) (Accompany Della0)) nil)))