プランナーと要求定義*

4L-3

中島 俊介　　橋本 正明†　　門田 充弘‡
ATR 通信システム研究所§

1 はじめに

プランナーは, 自分を取り巻く環境について問題意識を持ち, 問題が存在するような状況に陥った場合, それを解決するための行動を選んで実行する.

本稿では, システムをプランナーとして捉え, その要求定義を行う手法について考察する. 実例を通して, 定義項目の独立性, 仕様記述の再利用可能性を示す.

2 プランニング

プランニングの手続きと, その際に必要となる知識を示す. これらは文献[1] の枠組みを踏襲したものである.

プランニングの手続き

P1. 観測された事実により状況を更新.

P2. 現在の状況に問題があるかどうか調べる.
問題があるなら, 解決となるゴールを設定.

P3. ゴールを達成する可能性のあるアクションを提案.

P4. アクションを仮想的に実行.
状況の変化をシミュレート.

P5. すべてのゴールが達成され, 問題のない状況に至るまで P2-4 を繰り返す.

P6. 提案されたアクションの系列を実際に実行.

必要な知識

K1. テーマの定義.
(問題となる状況と解決となるゴールの対.)

K2. アクションの定義.

K2a. アクションを起こすために前提となる条件.

K2b. アクションを起こす前に成立し,
起こした後に成立しない事実.

K2c. アクションを起こす前に成立せず,
起こした後に成立する事実.

3 システムの記述例

在庫管理システムの注文 - 配達処理を例に, プランナーとしてのシステム記述を行う. 図 1 に状況の表現の例を示す. 状況は事実の集まりとして表現する. 図 2 にテーマ, 図 3 にアクションの定義を示す. 図 4 に入出力データの文法を示す. 入出力データは状況に対応付けされる.

*Planner and Requirements Definition
†Shunsuke NAKAJIMA, Masaaki HASHIMOTO,
‡Michihiro Monden
§ATR Communication Systems Research Laboratories

```
% X1 は X2 の一種である : kind_of(X1, X2).
% X1 は X2 の実例である : instance_of(X1, X2).
% X1 が X2 に入っている : 入っている (X1, X2).
% X1 の製造番号は X2 である : 製造番号 (X1, X2).

        [ kind_of(テレビ, 商品),
          instance_of(e1, 倉庫),
          instance_of(e8, テレビ),
          入っている (e8, e1),
          製造番号 (e8, 4193),
            ........   ].
```

図 1: 状況

```
% theme_type( <問題となる状況> ,
%              <解決となるゴール> ).

theme_type( [ 注文する (X1, X2) ] ,
            [ instance_of(X3, X2),
              配達する (X3, X1) ] ).
```

図 2: テーマ

図 2-4 により定義されたシステムは, プランニング手続きにより直接実行される. その流れについて説明する.

(1) 図 1 を現在の状況とする.

(2) データ“注文 佐藤 テレビ”を入力.

(3) 図 4 により入力データを解析.
事実『佐藤さんがテレビを注文する』を状況に追加.

(4) 図 3 により P1 を行う.
状況は変化しない.

(5) 図 2 により P2 を行う.
事実『佐藤さんがテレビを注文する』が状況に含まれている. これは問題となる状況である.
ゴール「 X3 を佐藤さんに配達する」を設定.
(ただし X3 はテレビの実例である何かである.)

(6) 図 3 により P3 を行う.

(6a) 状況からテレビの実例を捜す. e8 を得る.

(6b) アクション「 e8 を佐藤さんに配達する」の実行可能性を調べる.
事実『 e8 が e1 に入っている』が状況に含まれている. これは前提条件に反するので実行不可.

(6c) 状況から事実『 e8 が e1 に入っている』を削除する可能性のあるアクションを捜す.
アクション「 e8 を出庫する」を見つける.

(6d) アクション「 e8 を出庫する」の実行可能性を調べる. 実行可能.

```
% action_type( <アクション> ,
%              <前提条件> ,
%              <アクション以前の状況> ,
%              <アクション以後の状況> ).

% X1 が X2 を注文する.
action_type( 注文する (X1, X2),
             [ instance_of(X1, 顧客),
               kind_of(X2, 商品) ] ,
             [] ,
             [] ).

% X1 を出庫する.
action_type( 出庫する (X1),
             [ instance_of(X1, X2),
               kind_of(X2, 商品),
               instance_of(X3, 倉庫),
               入っている (X1, X3) ] ,
             [ 入っている (X1, X3) ] ,
             [] ).

% X1 を X2 に配達する.
action_type( 配達する (X1, X2),
             [ instance_of(X1, X3),
               kind_of(X3, 商品),
               instance_of(X2, 顧客),
               not 入っている (X1, X4)] ,
             [ instance_of(X1, X3) ] ,
             [] ).
```

図 3: アクション

```
% 入力 : 注文 <顧客名> <商品名>.
input(I1, I2, S) :-
        I1 = [注文, X1, X2 | I2],
        hold(S, 注文する (X1, X2)).

% 出力 : 配達 <顧客名> <商品名> <製造番号>.
output(O1, O2, S) :-
        O1 = [配達, X1, X2, X3 | O2],
        hold(S, 配達する (X4, X1)),
        hold(S, instance_of(X4, X2)),
        hold(S, 製造番号 (X4, X3)).
```

図 4: 入出力の文法

アクション「 e8 を出庫する」を提案.

(7) 図 3 により P4 を行う.
事実『 e8 が e1 に入っている』を状況から削除.

(8) 図 2 により P2 を行う. 新たな問題はないが, ゴールは達成されていない.
(過去に問題とされた状況は除いて考える.)

(9) 図 3 により P3 を行う.

(9a) アクション「 e8 を佐藤さんに配達する」の実行可能性を調べる. 実行可能.

アクション「 e8 を佐藤さんに配達する」を提案.

(10) 図 3 により P4 を行う.
事実『 e8 はテレビの実例である』を状況から削除.
(これにより e8 が認識外のものとなる.)

(11) 図 1 により P2 を行う. 問題なし, ゴールも達成されている.

(12) P6 を行う.
アクション「 e8 を出庫する」, 「 e8 を佐藤さんに配達する」を実行する.

(13) 図 4 により出力データを合成する.
データ“配達 佐藤 テレビ 4193”を得る.

(14) データ“配達 佐藤 テレビ 4193”を出力する.

4 要求仕様の記述

プランニング手続きを前提として, システムに対する要求の形式的定義を与えるには, 次に挙げる項目に関して記述を行う.

- テーマの定義.
- アクションの定義.
- 入出力の文法の定義.

ひとつひとつのテーマ, アクション, 入力の文法, 出力の文法の定義はシステムを取り巻く環境, すなわち実世界の状況に対して規定される. 従って, ひとつの定義を与える際に, 他の定義内容がどうであるか知る必要がなく, 独立に定義を与えることが可能である.

要求仕様記述のための状況の表現法には, 次の性質を備えた一般性の高いものが望まれる. (ここで言う表現法には, 単にどのような表現モデルを用いるかだけではなく, そのモデル上で様々な実在の概念をどのように表すかも含まれる.)

- 個々のシステムに依存しないこと.
- 人間による実世界の見方に近いこと.

表現法が個々のシステムに依存しなければ, あるシステムのために記述された個々の定義を, そのまま他のシステムの定義として再利用可能となる. 従って, ライブラリを設けることにより, システムごとの仕様記述量を少なくすることができる.

人間による実世界の見方に近ければ, それだけ仕様の記述や理解が容易になるし, システムを使う人間もまたプランナーと見なすことで, 人間や計算機を含めたトータルなシステムを扱うことができる.

5 おわりに

システムに対する要求をプランナーの定義として与えた. 本手法によれば, 要求は 3 つの項目, テーマ, アクション, および入出力の文法に対して定義される. いずれも実世界の状況との関係だけから規定されるので, すべての定義を独立に与えることができる. また, システムに依存しない状況の表現法を用いるなら, 個々の定義を再利用することが可能となる.

参考文献

[1] 岩瀬, 向井: 「状況意味論によるプランゴール理論の一考察」, 人工知能学会全国大会, 1987, pp.151-154.