ソフトウェア工学 89−19
（1992. 12. 16）

形式仕様記述言語G-LOTOSの記述試験

更科克幸† 安藤津芳† 太田正孝† 高橋薫‡

†高度通信システム研究所 ‡東北大学

あらまし 本論文では、具体的な交換システムを例題として、仕様の詳細化工程におけるG-LOTOSの記述試験を行った。LOTOSに於ける詳細化工程記述スタイルとしては、従来からのある構造化手法と今回取り上げた絞り込み手法が適応できた。絞り込み手法は、上位工程から下位工程へ連続的に行われる際に、下位仕様が記述すべき点を明示的に継承するため、要求モデルに沿って記述可能である。

Experiment with a Formal Description Technique: G-LOTOS

Katsuyuki Sarashina† Tsuyushi Ando† Masataka Ohta† Kaoru Takahashi‡

†AIC System Laboratories

6-6-3,MinamiYoshinari,Aoba-ku,Sendai-shi, Miyagi,989-32 JAPAN

‡Tohoku University

2-2-1,Katahira,Aoba-ku,Sendai-shi,Miyagi,980 JAPAN

ABSTRACT This paper reports on an experiment describing specification of a switching system using the graphical LOTOS and the narrow down methodology. The narrow down methodology enable to refined specifications smoothly according to requirement specifications.

1 はじめに

標準的なネットワークアーキテクチャの概念に基づいた通信規約の標準化がISO(国際標準化機構)や，CCITT(国際電信電話諮問委員会)で行われてきている．これらの規約は，殆どが自然語のスタイルを用いており曖昧性を完全に排除しきれない問題があった．そのため，同一の規約であっても，複数の解釈がなされたりして相互通信ができないなどという問題が表面化してきた．

そこで，意味的に複数の解釈がなされない形式的な言語を規定し，解釈に曖昧性が残らないようにすることを目的として形式記述技法(Formal Description Techniques)が標準化されてきている．現時点で標準化されているものとしては，CCITTでSDL[1]，ISOでは，Estelle[2]やLOTOS[3]が制定されており，その他にVDM，Zなど[4]の形式的記述方法も研究されている．

形式仕様記述技法を導入することによる利点は，文献[5]に記載されている．LOTOSについては，特に記述能力の高さ，検証能力の高さが言われている．しかし，LOTOSは数学モデルに基づいているために解読しにくく，必要性については認識されながらも現場になかなか普及しないのが現状である．更に，LOTOSは計算機による検証を行うためにテキスト型言語体系が主に利用されており，仕様書のドキュメント化，シミュレートに適している図式表現に対して，標準化が遅れていることも普及の鈍化を生んでいる．

本論文では，現在標準化作業中の，LOTOSのグラフィカルな表現(G-LOTOS)[6]を支援する記述環境[GLOER][7]を用い，仕様の段階的な詳細化記述を行う．そして，G-LOTOSの形式仕様記述技法としての実用性，将来性を模索し記述方法等を報告する．

2 G-LOTOS(LOTOS)の記述スタイル

LOTOSの記述スタイルは，ソフトウエア開発工程の流れに対して，その工程に最も必要で有効な記述スタイル取るべきとされ，表1に示す様に4つの記述スタイルが考案されている[9]．

それぞれの記述スタイルの具体的なG-LOTOSでの記述例を図1,2,3,4に示す．

●モノリシックスタイル(monolithic style)

モノリシックスタイルは，明確な全てのアクション(観測可能な動作のみ)を記述することを行いアクションのシーケンシャルな集合を記述する．特に，この記述スタイルは，仕様のデバッグ及び，テストシーケンスを分析する時に利用され，

表1. G-LOTOS(LOTOS)記述スタイル一覧

スタイル名	記述レベル	利点/欠点
モノリシック スタイル	要求仕様	直感的な理解性大 複雑な記述は不向き
制約指向 スタイル	要求仕様	制約の付加が容易 複雑な記述が可能
状態指向 スタイル	設計仕様	分野により理解性大 構造化は不可能
リソース指向 スタイル	設計仕様	部品単位の構造化可能 複雑な記述が可能

アクションは順序･分岐オペレーションを使った形で表現される．

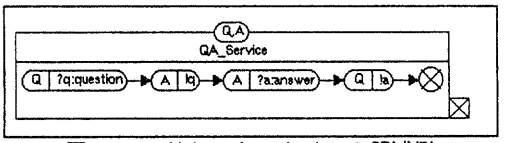

図1. モノリシックスタイルの記述例

●制約指向スタイル(constraint-oriented style)

制約指向スタイルは，マルチプロセッシング的な記述が可能な仕様記述技法のみ可能である．このスタイルは，プロセスが生起しうるアクションの集合列(順序集合)を，並列プロセス(関連動作集合)を用いてお互いの和集合･積集合をとる形で(制約する)記述方法である．

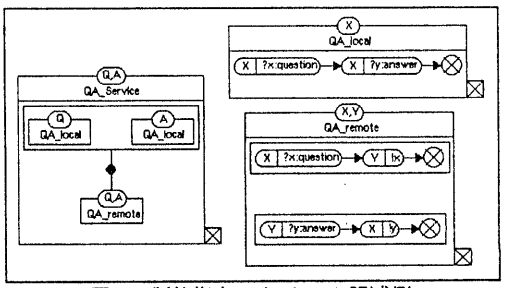

図2. 制約指向スタイルの記述例

●状態指向スタイル(state-oriented style)

状態指向スタイルは，状態値を用い状態の遷移毎のアクションを記述することによりシステムを記述してゆく方法である．アクションを状態遷移毎に分割するが，構造化が出来ないため，状態数が多くなると記述が巨大化する．

●リソース指向スタイル(resource-oriented style)

リソース指向スタイルは，仕様をリソース毎に分割する方法であり，仕様の詳細化に有効な記述方法である．具体的には1プロセスを1リソースに対応づけする事により記述がなされ，個々のリソー

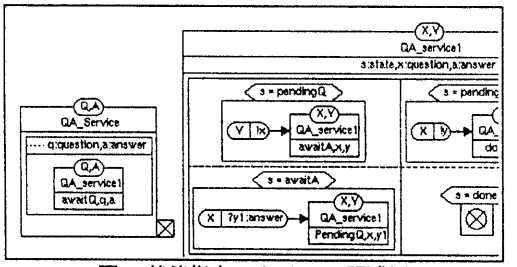

図3 状態指向スタイルの記述例

ス内の記述は，他の記述スタイル(モノリシックなど)が使われる．インプリメントレベルに近づく程に，この記述スタイルが用いられる．

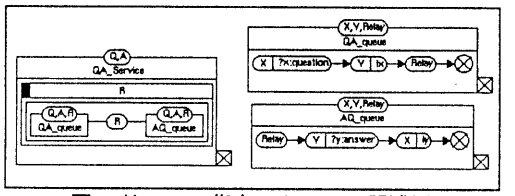

図4 リソース指向スタイルの記述例

上記で説明した記述スタイルの工程間での使い分けは，トップレベルの抽象的な要求仕様記述では，モノリシックや制約指向スタイルを利用する．そして，下位のレベルでは，リソーススタイルや状態指向スタイルにより記述してゆくことが望ましいと言われている[10]．

しかし，今まで説明した記述スタイルは，仕様化された時の表現方法であり，LOTOSを用いて仕様化を行う際の方法論について述べられたものは現状殆ど無い．

本論文では，4章で説明する記述対象モデルを使い，仕様の詳細化という観点から，G-LOTOSを利用した記述手法，記述手法から得られた記述試験結果を示す．

3 仕様の詳細化記述手法

仕様の詳細化とは，抽象的な仕様をなんらかの段階的な方法を使い，具体的な仕様へ変化させてゆくことを指している．プログラミング工程では，構造化プログラミング(構造化手法)が一般的に使われており，その他に，オブジェクト指向等が存在している．

仕様化工程でも，構造化手法を用いる事が可能である．しかし，この手法は仕様が構造化された結果，理解が容易であることに主眼が置かれており，仕様の作成工程には立ち入らない方法である．プログラミング工程では，仕様書をガイドラインとして構造化が行われるため，構造化作成工程がどの様に行われたかについては，それ程重要ではない．しかし，仕様化では具体的なガイドラインが無いため，仕様作成工程を明確にし，正確な具体化が行われたかを調べる必要性がある．そこで，仕様作成工程が分かる，絞り込み指向記述スタイルを提案する．この概念を説明するため，構造化手法と対比させ説明する．

構造化手法の例題を図5に示す．

抽象的な仕様に対し，リソースを割当て，そのリソースを記述すること(Step1)により，1段階具体化する．次々，上記の動作を繰り返しStep#n回目で記述された仕様から具体化なリソースが明示される．

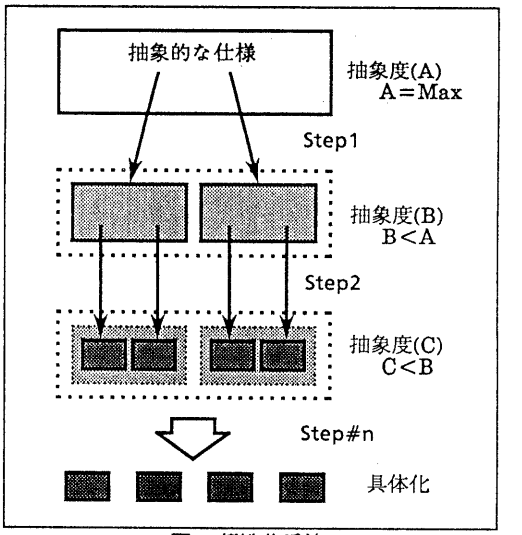

図5 構造化手法

絞り込み手法の例題を図6に示す．

抽象的な仕様に対し，明確な仕様部分を書き出し，抽象的な部分は，抽象化の仕様のまま残す方法である．図は，Step1で抽象的な仕様の側面が具体化され，まだ明確にならない部分はそのまま抽象化仕様のままであることを示している．そして，抽象的な仕様が全て，具体化された(Step#n)の時，具体化が終了する．この様に，抽象的な仕様を絞り込みながら段階的に具体化するので，絞り込み手法と呼ぶ．

構造化手法は，具体化されるレベルに於いて，リソース同志は同一レベルの具体化が成される．しかし，絞り込み手法場合，最終レベルに到達するまでは，具体化されたリソースと抽象的なリソースが混在した形をとる．次章で，具体的な仕様記述の例題を上げ，更に説明を加える．

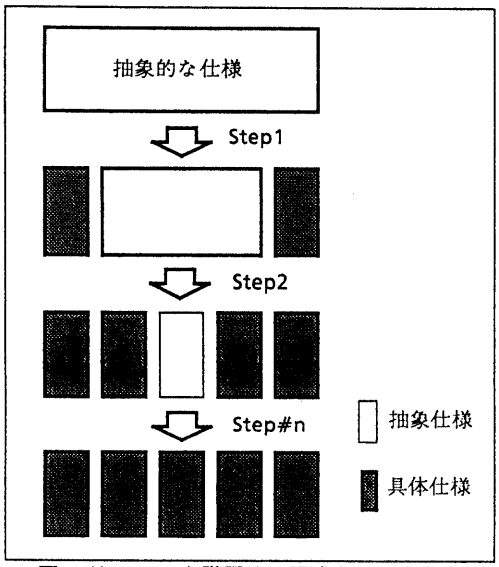

図6 リソースを階層的に記述する表現方法

4. 仕様記述試験

この章ではで，交換システムを例題に上げ，構造化手法と絞り込み手法の違いを明確にし，後半でG-LOTOSを用い実際に記述を行う．

要求仕様は，以下のように与える．

> 受話器を上げ、相手の数字を回す。すると相手に呼び出しベルが鳴り、こちらにも呼び出中の音が聞こえる。相手が受話器を上げることによって、通話が可能になる

図7に構造化手法を用い，具体化される交換システムの例で説明する．

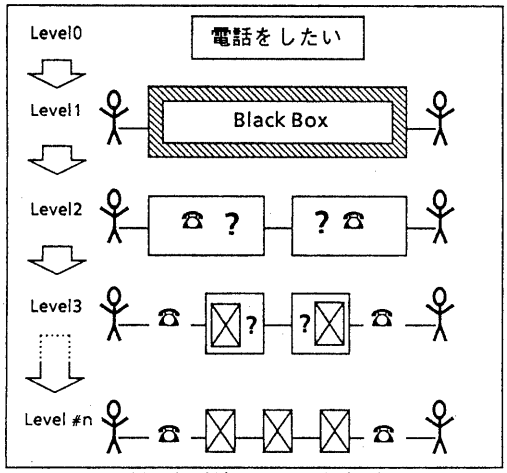

図7 構造化手法での例題解析手順図

各レベルの具体化は以下の様になる．

Level1 人ー電話機ー人のインタフェースの記述

Level2 人ー電話機ー電話機ー人のインターフェースの記述
(電話機同志が，接続されているように記述)

Level3 人ー電話機ー発信交換機ー着信交換機ー電話機ー人のインターフェースの記述(発･着信交換機が接続されているように記述)

アンダーラインの部分が図中の"?"の部分であり，実際のシステムでは矛盾している所である．しかし，次の具体化レベルで，この点が正確に修正されれば，結果としては矛盾は発生しない．

図8に絞り込み手法を用い，具体化される交換システムの例で説明する．先に提示した要求仕様から，各レベルの具体化は以下の様になる．

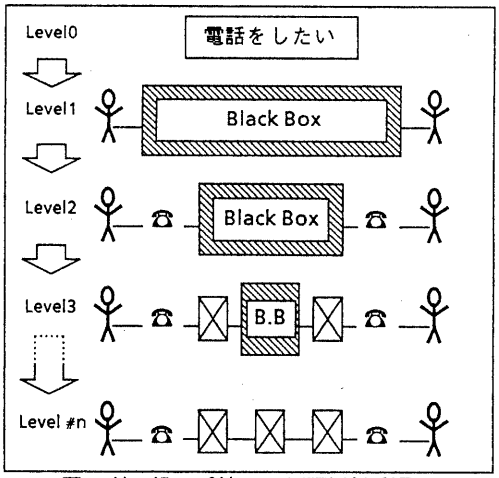

図8 絞り込み手法での例題解析手順図

Level1 人ー電話機ー人のインタフェースの記述

Level2 人ー電話機ー仮想網ー電話機ー人のインターフェースの記述

Level3 人ー電話機ー発信交換機ー仮想網ー着信交換機ー電話機ー人のインターフェースの記述

この記述方法では，構造化手法で発生した矛盾は起きなく，抽象的な仕様(仮想網)が明示される．
次の具体レベルでは，この抽象的な仕様を具体化すれば良いことが分かる．

G-LOTOSを使い，交換システムのLevel3までの仕様記述を行う．実際のインタフェース(信号シーケンス)を表したものが，図9である．

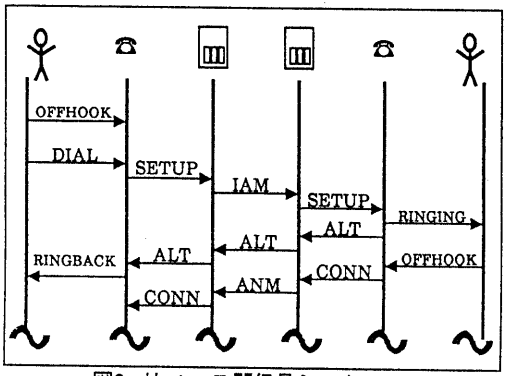

図9 リソース間信号シーケンス

Level1は，ユーザと電話端末間のシーケンス(受話器を上げる(OFFHOOK)，ダイヤルを回す(DIAL)，ベルが鳴る(RINGBACK, RINGING)などが記述される．Level2は，電話機と交換機間のシーケンス(SETUP, ALT, CONN)が記述される．Level3は，通信媒体(交換機)間のシーケンス(IAM, ACM, ALT, ANM)が記述される．

構造化手法で記述したG-LOTOS作図画面を図10に示す．Level3で記述された結果は，図9の示した信号シーケンスの縦線(プロセス)と図上のプロセスが一致し，リソースの並列性やインタラクションポイントが容易に読み取れる．

絞り込み手法で記述したG-LOTOS作図画面を図11に示す．この表現では，図8で示した詳細化の概念が読み取れ，どのような詳細化が行われたかを，階層構造により見ることができる．

5 記述評価

図10，図11を比較し評価を加える．

5.1リソース構造表現に対する評価

シミュレーション結果を含め，評価一覧を表2に示す．

①実システムとの対象性

実際のシステムは，各電話機，各交換機は全て並列に動作しており，並列動作のために発生する競合問題(シーケンス逆転等)が存在する．しかし，従来の仕様は，要求のみを記述しているためこのような問題点については現れない．仕様記述段階で，実システムで発生する問題点を認識できることが望ましい．

表2 詳細化工程の記述手法評価

	構造化手法	絞り込み手法
①実システムとの対象性	◎	○
②再利用性	X	◎
③拡張性	○	△
④視覚性	◎	○
⑤記述性	○	◎

構造化手法で記述した仕様は，リソースを並列に記述されるのでシミュレートの結果，上記の競合問題が再現でき現実的である．

絞り込み手法では，上位リソース自体が単一シーケンスで書かれているため，下位のプロセスが上位リソースの性質を継承する記述のため，並列動作自体は現れない．

②上位仕様の再利用性

詳細化工程における上位仕様から下位仕様への再利用性であるが，再利用性が高いことは，仕様記述の有効利用につながる．

構造化手法は，上位のリソースを下位のリソースへ完全に置き換えるため，再利用性はない．

絞り込み手法は，上位仕様に下位仕様を加える記述で済むため，上位仕様からの再利用性は高い．

③拡張性

リソースの変更･追加等の拡張性については，双方とも大差がない．

④理解性

G-LOTOS表現の直感的な理解性については，

構造化手法は，記述されたリソースの並列性が理解できる．最終的な仕様より，リソース数(構成プロセス数)も明確に理解できる．またインタラクションポイントの読解性も高い．

絞り込み手法は，システムの全体像などは，認識し難いが，階層構造見ることにより，詳細化工程の理解することができる．しかし，今回の記述では，詳細化で現れたリソースをLOTOSのプロセス表現に対応させていないので，見易くない．

⑤記述性

仕様の記述については，上位レベルからのスムーズな記述，仕様記述者の概念イメージのスムーズな具体化が重要である．

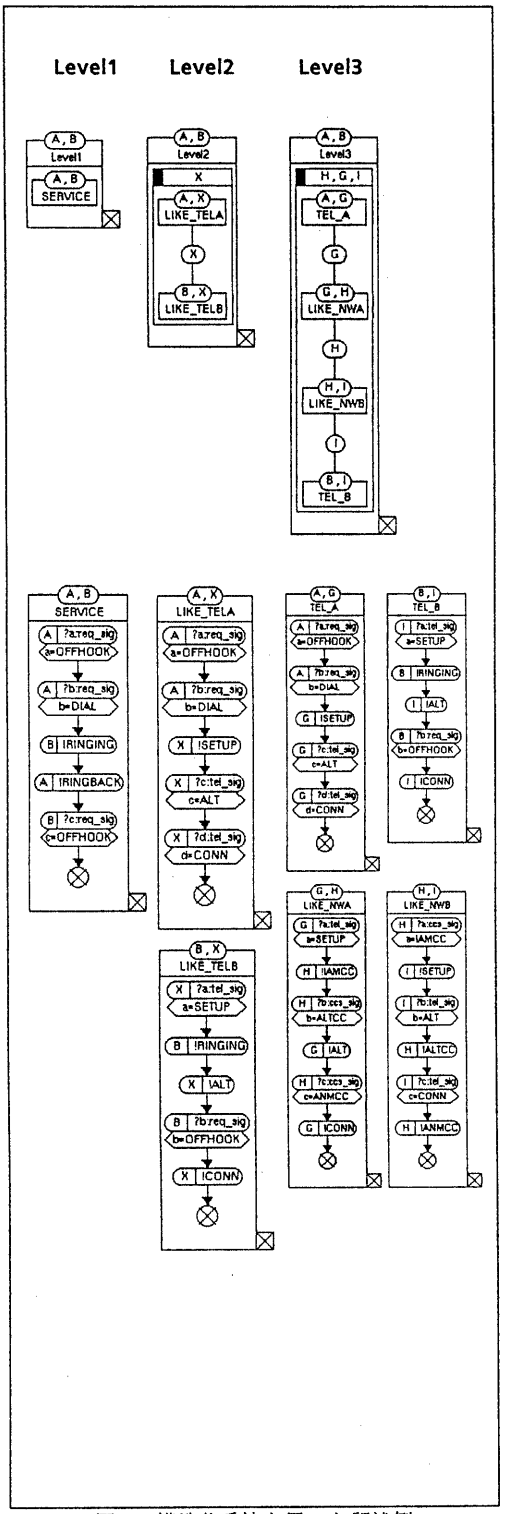

図10 構造化手法を用いた記述例

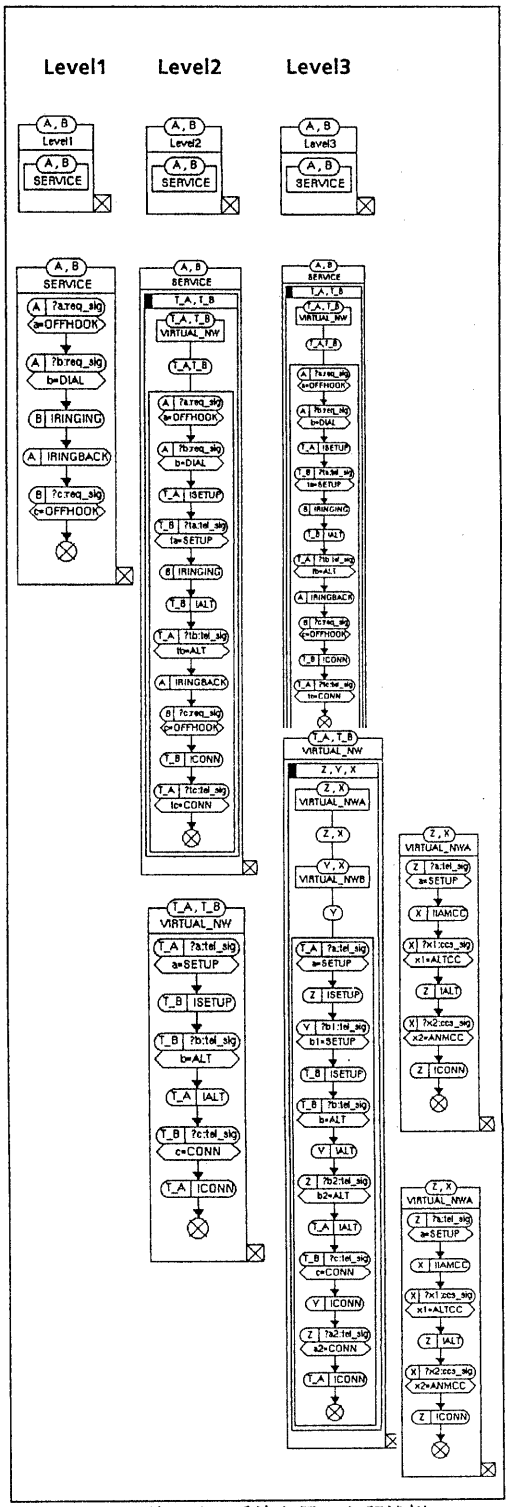

図11 絞り込み手法を用いた記述例

構造化手法は，上位仕様から下位仕様への具体的な記述面での引継ぎが無いため，本来の要求仕様の枠組みを越える恐れがある．しかし，個々のリソースを分散して詳細化記述をすることもできる．

絞り込み手法は，上位仕様から下位仕様への引継ぎが明確なため，本来の仕様の枠組みを越える危険性は少ない．記述量は，構造化手法の方が，上位仕様の引継ぎ仕様を描く必要がないため少ない．

仕様の詳細化工程に於ける，2つの詳細化記述手法について比較した．構造化手法は，抽象仕様(上位仕様)から具体仕様(下位仕様)へ正確なインプリメントが行われるかが問題である．この点から考えると，上位仕様から次に何を記述すれば良いかを下位工程に引継ぐ方法の，絞り込み手法の方が勝っている．しかし，仕様のシミュレート時点で，アクション競合等の問題点が認識可能な構造化手法についても無視できない利点である．

5.2個々のリソースに表現の評価

リソース(プロセス)内部を記述するスタイルのモノリシック，状態指向，制約指向の3つを実際記述した例が図12である．これらの記述スタイルについて，表3に評価を示す．

表3 リソース自体の記述評価

	モノリシックスタイル	状態指向スタイル	制約指向スタイル
①視覚性	◎	○	△
②記述性	◎	○	◎
③変更の容易性	△	◎	○

①視覚性

モノリシックスタイルは，アクションの直感的な理解性が良く，アクションの順序性などは，一目瞭然である．

状態指向スタイルは，状態によるアクション系列の記述が特徴であるが，状態遷移を分岐オペレーションで記述するため，状態の順序性については，理解し難い．

制約指向スタイルは，並列プロセスの記述となるため，理解性が決して良いとは言えない．しかし，アクションの順序，入出力アクション関係を，別プロセスに書くため奇麗である．

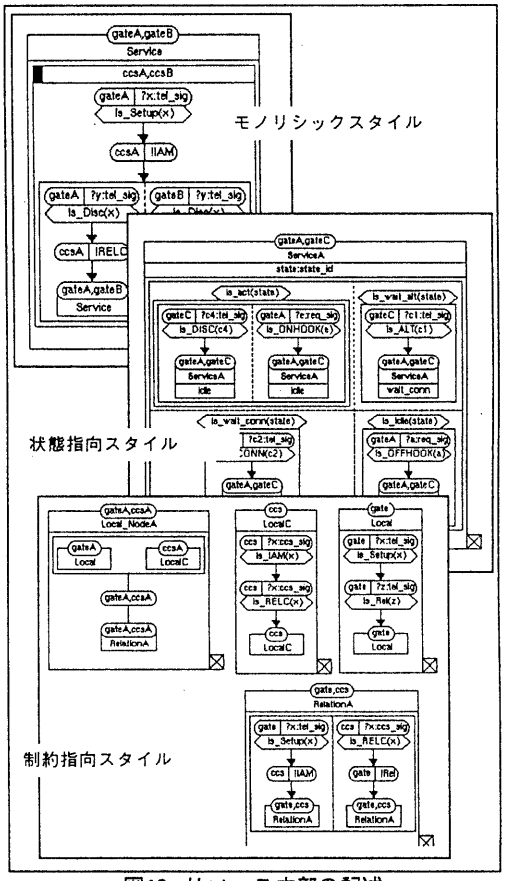

図12 リソース内部の記述

②記述性

モノリシックスタイルは，生起されるアクションのみをアクションの生起順序通りに記述するので，記述は容易である．

状態指向スタイルは，状態マシンに対する十分な経験･知識があれば，容易に記述できる．

制約指向スタイルは，アクション順序列の記述，アクション関係の記述が分かれているため，奇麗に記述できる．

③変更の容易性

モノリシックスタイルは，変更が仕様全体に及ぶため，変更に伴う仕様の正常性チェックは，既存部分まで必要であり容易とは言えない．

状態指向スタイルは，仕様の変更がどの状態に及ぶかを見極める事が重要なポイントであるが，変更は比較的容易である．修正後の仕様の正常性チェックは，変更が行われた状態の記述部分に限定できる．

制約指向スタイルは，仕様の変更に対して制約を変更することにより対処可能である．修正後の仕様の正常性チェックについても，状態指向スタイルと同様に変更が行われた部分に限定できる．

モノリシックの場合は，アクションが，木構造的な広がりで記述されるので，アクションが少ない場合，非常に理解しやすい利点が見られる．しかし，アクションが多くなり，複雑な記述になると読み難く，仕様の部分的変更が記述全体に影響する難点がある．

状態指向の場合は，現状態から次状態までを独立プロセスとして置き換えれるため，仕様の変更改良については，容易に行うことが可能である．しかし，状態マシンを扱うため，仕様記述者に充分な知識が必要である．

制約指向の場合は，記述の方法が理解できれば，特に状態指向スタイルで必要な知識が無くても記述ができる．しかし，制約を並列プロセスで記述するためにプロセスがさらに階層化されるため，リソース指向スタイルでの階層化と意味的に異なる階層が現れ，読み難くなる．

6 結論(まとめ)

具体的な交換システムを例題として，仕様の詳細化工程におけるG-LOTOSの記述試験を行った．LOTOSに於ける詳細化工程記述スタイルとしては，従来からのある構造化手法と，今回取り上げた絞り込み手法が適応できた．

絞り込み手法は，上位工程から下位工程へ連続的に行われる際に，下位仕様が記述すべき点を明示的に継承するため，要求モデルに沿って記述できる．

構造化手法は，実システムの競合問題が仕様記述段階で現れることが分かり，実システムを正確にリソースへの置き換えができれば，仕様段階でのデバッグか可能であることが分かった．

今後は，それぞれの詳細化記述技法の利点を融合させる方法の検討，GLOERでの上記記述手法の機械的サポート方法を検討する．

謝辞

本研究を進めるにあたり，御指導，御支援を頂いた東北大学の野口正一教授，ならびに白鳥則郎教授に深謝致します．また，研究をする機会を与えてくださった同研究所の緒方秀夫常務，有益な御意見を頂いた同研究所の諸氏に感謝致します．

参考文献

[1] CCITT:"Functional Specification and Description Language(SDL)",Recommendation Z100, 1989

[2] ISO:"Information processing systems-Open Systems Interconnection -Estelle- A Formal Description Technique Based on an extended state transition model", ISO9074, 1989

[3] ISO:"Information processing systems-Open Systems Interconnection -LOTOS- A Formal Description Technique Based on Temporal Ordering of Observational Behaviour", ISO8807, 1989

[4] Springer-Verlag:"VDM'90 VDM and Z Formal Methods in Software Development", ThirdInternational Symposium of VDM Europe, 1990

[5] 二木厚吉:"ISOにおける形式記述技法の標準化動向",情報処理, Vol31, No.1, 1990

[6] ISO:"Revised Text of ISO 8807/PDAM 1, Information Processing systems-Open Systems Interconnection -LOTOS- A Formal Description Technique Based on Temporal Ordering of Observational Behavior - Draft Amendment 1: G-LOTOS", ISO/IEC JTC 1/SC 21 N 6751, 1992

[7] 更科克幸,山野敬一郎,大友弥生,安藤津芳,太田正孝,高橋薫:"GLOER:G-LOTOSエディタの試作",情報処理学会第42回全国大会,1991

[8] 高橋薫,神長裕明,白鳥則朗:"LOTOS言語の特質と処理系の現状と動向", 情報処理, Vol.31, No.1, 1990

[9] C.A.Vissers,G.Scollo and M.V.Sinderen: "Architecture and Specification Style Formal Descriptions of Distributed Systems",IFIP Protocol Specification testing and Verification VIII, North-Holland,1988

[10] 大蒔和仁,二木厚吉,高橋薫: "LOTOS-実用指向のプロセス代数(応用編)", Bit , 1992