知識ベース利用の SDL 支援システム†

藤本 洋†† 鈴木忠道††† 加藤英樹††††

電子交換機に要求される交換サービスの多様化に伴い，そのソフトウェアの規模は大型化の一途をたどりつつある．この傾向は，ソフトウェア開発費・保守費の増加を招き，その対応策として，飛躍的な生産性の向上のみならず，信頼性・保守性をも向上させることが重要な課題となってきている．本稿は，このような背景から，交換ソフトウェア仕様からのプログラム自動生成という課題に取り組んだものである．この自動プログラミングは，従来より国内・海外共に幅広く研究・開発されているが，いまだ小規模プログラムを生成する程度，もしくは入力する仕様と生成されるプログラムの記述レベルが，ほぼ1対1対応の生成機能のものが多い．本稿では，仕様とプログラム間のセマンティックギャップを克服するため，自動生成過程の抽象化レベルを2段階に分け，ここに知識処理技法を導入することによって，通信ソフトウェアという限られた分野ではあるが，自動プログラミング実用化の可能性が明らかにできたことを示す．試作システムの開発には，仕様記述言語として CCITT（国際電信電話諮問委員会）勧告の SDL を採用し，交換ソフトウェアの呼処理部分を生成対象としている．

1. まえがき

交換機の動作を論理モデル化すると，電話端末の操作や対向する交換機との中継接続動作に伴う外部信号によって，システムの次の状態が決定される一種の状態遷移マシンと考えることができる．したがって，交換ソフトウェアの設計作業は，まず交換サービス仕様を満足するように，交換機の外部信号に応じた状態遷移を定義することから始める．さらに，各々の状態遷移動作に必要な手順を明確にしながら，交換サービスを直接処理する呼処理部分を中心に，全体的なソフトウェア設計が行われる．

最大許容加入者数が 10 万を越える大規模ディジタル局用交換機や提供サービス種別が数百にも達する複合 PBX の場合，交換ソフトウェアは100万ステップにも及ぶプログラムで構成され，必要な開発要員は数百名にも上っている．また，開発コストに占めるソフトウェアのコストも年々増加し，特に機能追加・保守に要するコストが著しく増大する傾向にある．

このような状況の中で，交換ソフトウェア開発の効率化，特に高度な技術・経験を持った設計要員不足の解消を図ると共に，設計工程での品質向上による試験工数の削減を図ることが不可欠な課題となっている．とりわけ，上流設計工程での作業の効率化と設計品質の向上に関しては，ほとんど未開拓の課題であり，本格的に取り組む必要がある．我々は，この課題に対して，仕様からのプログラム自動生成を中心に設計作業を一貫して支援する，交換ソフトウェアの設計支援システムの構築に取り組んでいる．

本稿では，CCITT で勧告されている通信ソフトウェア向け仕様記述言語 SDL (CCITT Specification and Description Language) の言語的特質を利用し，仕様からプログラム生成の過程での抽象化レベルの段階分け技術と人工知能分野で研究されている知識処理技術[1]の統合により，交換ソフトウェア呼処理部分の自動プログラミング実用化の可能性を明らかにすると共に，プログラム生成機能を中心としたソフトウェア設計支援システム SKBS (SDL Knowledge Based System) の構成概念について述べる．

2. SDL の概要

SDL[2] は，『電話交換機に代表される通信システムの仕様やプロトコルを正確に記述する』ことを当初の課題として，CCITT (The International Telegraph and Telephone Consultative Committee：国際電信電話諮問委員会) で研究・開発されている，通信ソフトウェア向け機能仕様記述言語である．

(1) 経　緯

ソフトウェア・エンジニアリング技術に関する研究・開発開始と同時期の 1968 年に研究が開始され，段階的な勧告を経て 1988 年に最終勧告される予定で

† SDL Support System Using Knowledge Base by HIROSHI FUJIMOTO (Telecommunications Software Development Division, Fujitsu Limited), TADAMICHI SUZUKI (Engineering Department, Telecommunications Software Development Division, Fujitsu Limited) and HIDEKI KATOH (Artificial Intelligence Laboratory, Fujitsu Laboratories Ltd.).

†† 富士通(株)通信事業推進本部ソフトウェア開発部
††† 富士通(株)通信事業推進本部ソフトウェア開発部ソフトウェア技術部
†††† (株)富士通研究所人工知能研究部

ある.

- 1976年：図形表記法 SDL/GR の制定
  (SDL Graphical Representation)
- 1980年：絵素の追加，文字テキスト表記法 SDL/PR の制定 (SDL textual Phrase Representation)
- 1984年：抽象データ型，階層的定義記法の追加

(図 1，図 2)

(2) 特　　徴

『通信システムの機能的な振る舞いを，曖昧なく記述でき，形式的な解析を可能とする』ことを狙いに開発された SDL の特徴は以下のように要約できる.

- 図形表記 (SDL/GR) と文字テキスト表記 (SDL/PR) により，マン・マシンの両者に対応できる.
- 状態遷移による機能動作表現を用いて，機能的な振る舞いを記述できる.
- 任意の抽象度表現ができるため，定義語句に対して任意に意味付けが可能であり，文脈依存で任意の文章構成が可能である.

(3) 評　　価

上記の特徴を有する SDL は，交換ソフトウェア設計支援の立場からは，以下のように評価できる.

- 仕様記述能力：交換機の動作は有限状態遷移マシンとして論理モデル化できることから，状態遷移による動作表現機能を有する SDL は，交換ソフトウェアの仕様を簡潔に記述できる.
- 設計記述能力：SDL は，要求定義から基本設計工程での仕様の詳細化記述が可能であり，任意の抽象化レベルで記述し，さらにオプショナルな表記を追加することにより，交換サービスを直接処理する呼処理部ソフトウェアの論理的な動作を記述できる.
- ソフトウェア CAD (Computer Aided Design) 適合性：図形表記 (SDL/GR) と文字テキスト表記 (SDL/PR) を使い分けることにより，高度なマンマシン・インタフェースを持った使い勝手の良いソフトウェア CAD の構築が可能である．すなわち，図形による人間の内容理解と文字テキストによる内容の蓄積・検索・加工の機械処理が容易にできる.

3. SDL ベースのソフトウェア設計

仕様からプログラムの自動生成を図るソフトウェア設計支援システムを構築するために，仕様とプログラムとのセマンティックギャップを埋める手段として，SDL の抽象化レベルの段階分けが重要な技術的ポイントと考える．そのアプローチとして，交換ソフトウェア設計の基本的特徴，特に設計工程区分に着目した SDL の適用方針を定めた上で，その段階分けを行うこととした.

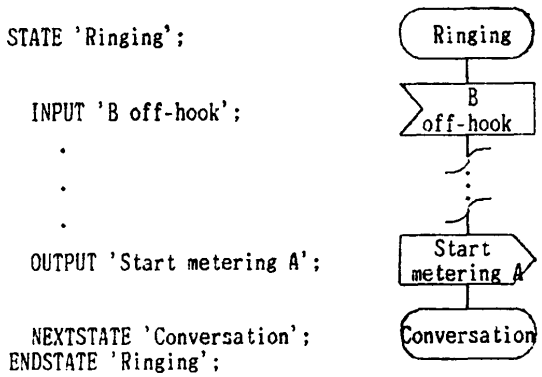


図 1 SDL/PR と SDL/GR の対応
Fig. 1 Correspondence between SDL/PR and SDL/GR.

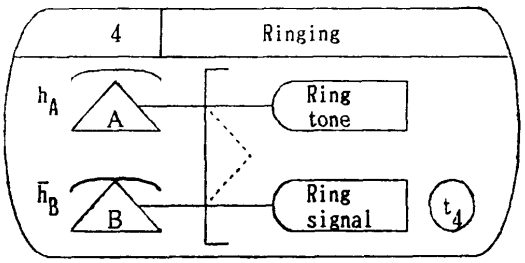


図 2 状態記号内に絵素を使った例
Fig. 2 Example of pictorial element described in a state box.

(1) 交換ソフトウェアの設計

交換ソフトウェアの設計は，①システム要求定義工程で決定された交換機のサービス仕様を受けて，②システムの系構成条件等を基にソフトウェアとしての仕様の詳細化を行うソフトウェア要求定義工程，③ソフトウェア仕様からソフトウェアの基本的な動作を決定し，ソフトウェア構成を決定する基本設計工程を経て，分割された各々のソフトウェア部品のアルゴリズムを決定し，プログラムコード化する工程に至る.

特に，基本設計工程において，交換サービスを直接処理する呼処理部分のソフトウェア基本動作の決定を先行させながら，各々の加入者線ごとにサービス条件の変更等運転状況を監視・制御する保守運用ソフトウェア等の付帯的なソフトウェアの基本動作の決定と各々のソフトウェア構成の決定を並行的に行っていくことが，交換ソフトウェア設計の大きな特徴である.

（2） SDL の適用方針と抽象化レベルの段階分け

上記の特徴を踏まえ，設計のキーポイントとなる呼処理部分の設計上位3工程に関して，「What」と「How」の区別を対応付けると，SDL の記述レベルは次のとおりに分けられる．

① S-SDL : SDL for Service description
システムとして「何をすべきか」，交換サービス仕様の記述（システムの「What」）

② G-SDL : SDL for General behavior description
交換サービスを直接処理するために，呼処理部分のソフトウェアが「何をすべきか」の記述（ソフトウェアの「What」）

③ D-SDL : SDL for Detailed behavior description
呼処理部分のソフトウェアが「いかにすべきか」，動作の記述（ソフトウェアの「How」）

以上の考え方で，各レベルの SDL で記述すべき項目を，**表1**のように決定すると共に，本稿では G-SDL と D-SDL の2段階の抽象化レベルでのプログラムの自動生成を図った．

4. 設計支援システムの開発思想

プログラム生成の入力手段として，特に G-SDL を適用し，D-SDL を経てコード生成する設計支援システムの開発において，導入した技術的ポイントを示す．

（1） 知識のフレームへの変換

プログラム生成機能を考慮すると，入力される SDL を画素情報としてでなく，SDL/GR のシンボルごとの構成情報として処理する必要がある．本稿では各サブシステムに共通な内部データ構造として，知識表現用の汎用データ構造であるフレームを用いた．したがって，共通なデータは上位レベルからインヘリット (Inherit) され，データの書き換えに伴う関連データの更新はデモン (Demon) が行う．その結果，データ構造全体の一貫性，整合性が自動的に保たれ，プログラム側の負担をかなり軽減できる．

（2） 知識負担の最小化

一般に交換サービス仕様には，サービス種別・信号方式・タイミング条件・課金方式・ハードウェアリソース等の情報があり，設計者はこの中の記号列から，機能的な「意味」を理解した上で，プログラミング言語に変換している．従来の仕様記述言語からの自動プログラミング技術として，『多様な交換サービスを対象とするには，記述し得る情報の自由度を制限せざるを得なく，逆にその自由度を持たせることに重点をおくと，処理系で持つべき変換パタンが膨大なものになる』という欠点があった．換言すれば，仕様記述における抽象化レベルの設定法次第で，設計者の知識負担の度合いが大きく異なることになる．

本稿では，交換サービス仕様に関する知識を，支援システムに内包することにより，各手順での設計者の知識負担を最小化する仕組みを実現する．

① 〔手順1〕 既存のソフト資産を再利用しつつ，新しいサービス仕様を基本設計する．⇦SDL 編集サブシステムによる支援⇦SDL の記法に関する知識

② 〔手順2〕 仕様を理解し，プログラミングできるレベルまで詳細化する．⇦タスク生成による支援⇦概要仕様の記述用語および詳細仕様の記述用語に関する

表 1 各レベルの SDL の記述項目
Table 1 Description contents of SDL at each level of abstraction.

言語レベル		S-SDL	G-SDL	D-SDL
記述内容		提供する交換サービスの各々（自局内接続，入接続，出接続等）について，交換機の一連の状態遷移の流れを記述する．	次の状態へ遷移させるための交換機の動作を，交換機を構成するサブシステムごとに記述する．	ソフトウェアの動作を盛り込んで，サブシステムごとの状態遷移動作を記述する．
使用シンボルとその意味付け	状態	●状態名	●状態名 ●交換機全体の状態	●状態名 ●サブシステムの状態
	入力	●外部入力信号	●サブシステム入力信号（含む，外部信号）	●サブシステム入力データ（入力信号詳細化）
	出力	●外部出力信号	●サブシステム出力信号（含む，外部信号）	●サブシステム出力データ（出力信号詳細化）
	判断	●次状態の選択判断（外部条件）	●次状態の選択判断（システム条件の追加）	●次状態の詳細選択判断（ソフト処理条件の追加）
	タスク			●ソフトウェアの処理動作（データの検索，加工等）

知識

③ 〔手順3〕 使用するプログラミング言語でコーディングする．⇐コード生成による支援⇐対象システムで決まっているプログラミング規則に関する知識

（3） 知識ベースの構築

設計者が持っている断片的な知識を，次の3点に着目してルール形式で記述し，知識ベースを作成する．

① SDLで記述される記号列から，本質的で，かつ交換システム種別に依存しない普遍的な「意味」を抽出し，それを知識ベース（G-SDL要素知識）として格納する．

② 普遍的な「意味」から，ソフト系構成を踏まえて「いかにすべきか」を整理し，それを知識ベース（D-SDL要素知識）化する．

③ 「意味」を次の3つのパラメータによって抽出する．

：名前，条件，作用

すなわち，SDLで記述される表現（名前とかシンボル種別とか）が何を意味するか．それが指定されたら，次に何をしないと“交換システムとして”正しくないか等を，整理された名前ごとに登録する．

（4） プログラム生成の最適化

SDLは仕様記述言語であり，使い方の上ではプログラミング言語と大きく異なる．CCITTの勧告では，理想的なアーキテクチャ（例えば，プロセスは相互に完全独立，仕様—プログラム間は完全にレイヤ分離，等）を想定しているが，現段階では性能条件の制約から上記の理想的な切り分けは困難である．したがってSDLを適用する場合にも必然的に何らかの工夫が必要となる．本稿では，コード生成部の省略・文脈処理において，最適なコード（例えば，受信信号を分析・判断する場合の特殊な変数名およびフィールド名，仕様記述上省略される固定的なタスクマクロ名，等）を生成できるよう，最適化ルール関数を用意している．

（5） テキストと図形の相互変換

SDL/GRによる記述をSDL/PRに変換してファイルに格納したり，格納されたデータをSDL/GRに戻して，その上に変更，追加等を行えるよう，①図形情報の自動付加，②配置支援機能を持つ．

5. 支援システム構成概要

本システムSKBS[3)~6)]は，4つのサブシステムから構成される．設計者はSDL編集サブシステムによって，新しいサービス仕様に基づくソフトウェア要求定義を行う．同時に，SDL検証サブシステムによって，仕様レベルの誤りを除去する．その後，タスク生成サブシステム，コード生成サブシステムを用いて詳細化を行う．これらのサブシステムは知識ベースを中心に，交換ソフトウェア設計の一貫支援システムとして構成されている．次に各サブシステムについて述べる（図3）．

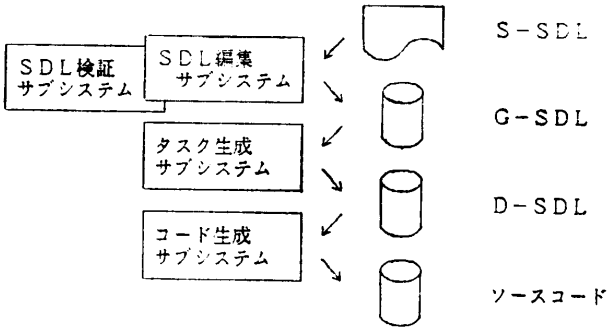


図3 言語レベルと変換過程

Fig. 3 Relation between the language level and the transformation process.

5.1 SDL編集サブシステム

5.1.1 サブシステム構成概要

本サブシステムは，SDL/GRによる仕様記述作業を支援するもので，設計者からのコマンドを解釈実行するコマンドインタプリタを中心に，図形端末や日本語ラインプリンタ上に描かれるSDL/GRとファイル上に格納されるSDL/PRとの相互変換を行うための表記コンバータ，ファイル管理ルーチン，グラフィックルーチンとで構成される．

5.1.2 特　　徴

本サブシステムは以下の特徴を持つ．

（1） メニュー方式の採用

仕様記述のための基本的な操作は画面上のメニューを選択することにより実行できる．

（2） 対話型実行システム

情報不足や曖昧さがある場合でも，システムからの質問に答える形で，編集作業が行える．

（3） 画面位置情報のフレームデータベース管理の採用

SDL/PRにはSDL/GRに変換する際の位置情報が含まれないため，SDL/GR編集時に画面位置情報を自動的に収集し，SDL/PR変換時にその位置情報をフレームデータベース化して別ファイルに格納してい

る．これにより，図形表示時の座標位置変更の追随性を高めている．

（4） 更新履歴管理機能の具備

SDL/PR をファイル管理し，更新履歴機能を具備することにより，データの保全と世代管理が容易になり，任意の旧版に戻すことも可能となることから，他システムへの設計生産物の再利用も容易となる．

5.2 タスク生成サブシステム

5.2.1 サブシステム構成概要

本サブシステムは，G-SDL による仕様記述から，その仕様を満足する D-SDL を生成する．**図 4** にタスク生成の過程を示す．知識ベースには，G-SDL，D-SDL それぞれの要素知識が格納されている．フレーム変換部は，「意味」を抽出する部分で，入力された G-SDL 中の記号ごとに，その記号種別・名前をキーとした知識ベースのパターンマッチング検索を行い，対象システムの作用を一般的な記述に変換する．次に，推論部は，「詳細化」する部分で，この変換された記述を達成目標として設定し，その目標を達成できる D-SDL の要素知識を知識ベースから探し出す．通常，探し出された要素には，さらに達成すべき要件（要素知識）が書かれているので，その要件を達成目標として，知識ベースの検索を続ける．すべての要件が満たされたとき，その要素知識に関する変換が完了したとみなす．

5.2.2 知識ベースの構成方式

タスク生成に必要な知識は，入力 G-SDL に出現し得る要素知識の定義と，出力 D-SDL 要素知識の定義である．特に作用の定義形式は，

＜作用＞：：＝（＜動詞＞＜目的語＞）

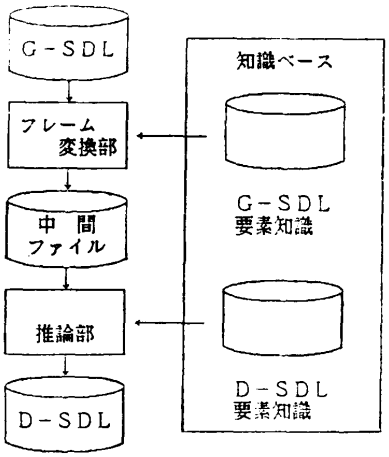


図 4 タスク生成過程
Fig. 4 Task generation process.

で記述される．設計者の知識を知識ベースに格納する過程を，簡単な例で説明する．

- 加入者Aが受話器を上げる行為は，システム内に起こる1事象で，入力 A off-hook で表現される．
- 加入者Bが受話器を上げる行為も，システム内に起こる1事象で，入力 B off-hook で表現される．
- 同様なことが，加入者のすべてに当てはまる．

これを一般化すると次のようになる．

- 1加入者を変数 X で表すと，「加入者 X が受話器を上げる」行為は，システム内に起こる1事象で，入力 X off-hook で表現される．

したがって，専門家が A off-hook という記号に出会うと，X off-hook という一般化された知識が頭に浮かび，さらに A は変数 X に代入された値であることを認識する．この一般化された知識は，名前，条件，作用の3属性に分けて，下記に示す形で知識ベースに格納されている．

属性	記述
名前	X off-hook
条件	X は加入者名
作用	加入者 X が受話器を上げる．

上記例の知識は，次の事実を示す．

：X を加入者とするとき（条件），入力記号 X off-hook（名前）は，システムの内部状態に与える作用の面から見ると，「加入者 X が受話器を上げる」というシステム事象（作用）が起こることを意味する．

以上は G-SDL に関する知識の例である．

5.2.3 タスク生成動作概要

（1） フレーム変換部

G-SDL の記号からの変換過程を，**図 5** に示す G-SDL が入力されたものとして説明する．

① A off-hook を一般化した X off-hook を名前とする要素知識を，入力フレームの中から探し，そ

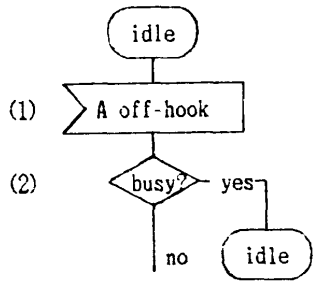


図 5 入力される G-SDL
Fig. 5 General SDL to be inputted.

の要素の作用欄の記述を，中間ファイルに入れる．

② busy? を名前とする要素知識を，判断フレームの中から探し出し，その作用欄の記述を，中間ファイルに入れる．

この結果，中間ファイルには，以下の記述が格納されている．

(i) 入力 加入者 X が受話器を上げる．

(ii) 判断 (OR (NOT 加入者用作業域の確保成功)
(NOT トランクの確保成功))

(2) 推論部

D-SDL に関する知識の場合には，名前，作用のほかに，次の3つのスロットが追加されている．

前処理：要素実行に必要な前処理

後処理：要素実行に必要な後処理

判断結果：yes および no の分岐先である判断結果フレームへの2つのポインタ

以下は D-SDL の記号に関する知識の例である．

種別	属性	記述
入力	名前	X off-hook from LC
	作用	加入者 X が受話器を上げる
	後処理	X の補助情報を設定する

次に，以下の内容が格納されている中間ファイルを，D-SDL 要素列に変換する過程を示す．

① 入力 加入者 X が受話器を上げる

② 判断 (OR (NOT 加入者用作業域の確保成功)
(NOT トランクの確保成功))

まず，①の「加入者が受話器を上げる」という作用を達成できる要素を求めて，入力フレームをサーチする．作用が一致する要素が見つかると，その名前 X off-hook from LC に X=A を代入して，出力要素列に挿入(a)する．

この要素には後処理として，「X の補助情報を設定する」とあるので，これを目標として知識ベースを再びサーチする．タスクフレーム中に，目標を達成できる要素を見つけ，その名前 get detail on X に X=A を代入して，出力要素列中の A off-hook from LC の直後に挿入(b)する．これによって，①の入力記号に対する処理を完了する．

ここまでの変換過程を，**図6**に示す．図中，実線による矢印は，処理の順序を示し，点線は，出力要素列への挿入を示す．

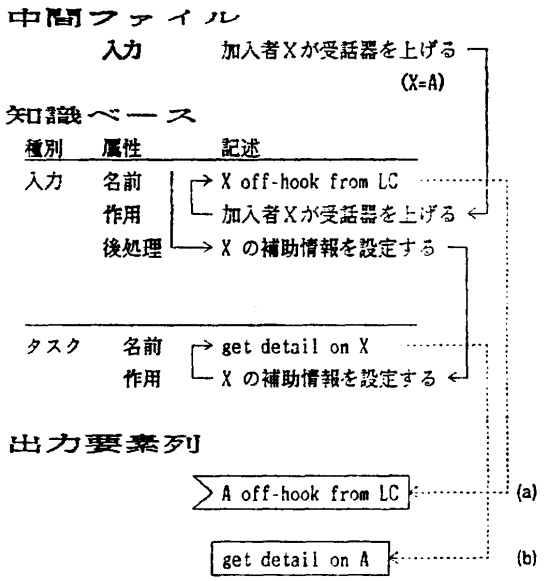


図6 推論過程 (1)
Fig. 6 Inference process (1).

次に，②の (OR (NOT 加入者用作業域の確保成功) (NOT トランクの確保成功)) を作用として持つ要素をサーチするが，そのような要素は存在しないので，このサーチは失敗に終わる．この場合，対象要素が判断記号であり，その作用の記述である論理式は，(OR (NOT A) (NOT B)) の形をしているので，その構成要素AおよびBをそれぞれ単独の目標として，目標達成可能な要素をサーチする．

まず，A「加入者用作業域の確保成功」を目標として判断フレームをサーチし，所要の要素を見つけ出し，その名前 request OK? を出力要素列に挿入(c)する (**図7**)．

見つかった要素には前処理の記述があるので，その作用に対する記述「加入者用作業域を要求する」を新たな目標として，知識ベースをサーチする．その結果タスクフレームの中に目的の要素を見つけると，その名前 request work area を request OK? の直前に挿入(d)する．

さらに，判断結果の ADDR (yes) の指す判断結果フレームを参照する．作用スロットを持たないので，yes 分岐に対する処理は終わる．

今度は，ADDR (no) の指す判断フレームを見ると，後処理として「busy tone を送出する」があるので，これを目標として，知識ベースをサーチする．

タスクフレーム中に所要の要素を見つけ，その名前 send busy tone を出力要素列に挿入(e)する．

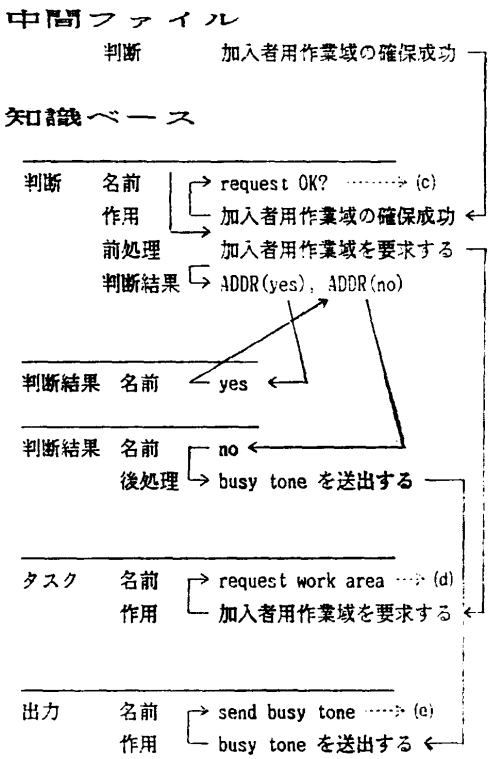


図 7 推論過程 (2)
Fig. 7 Inference process (2).

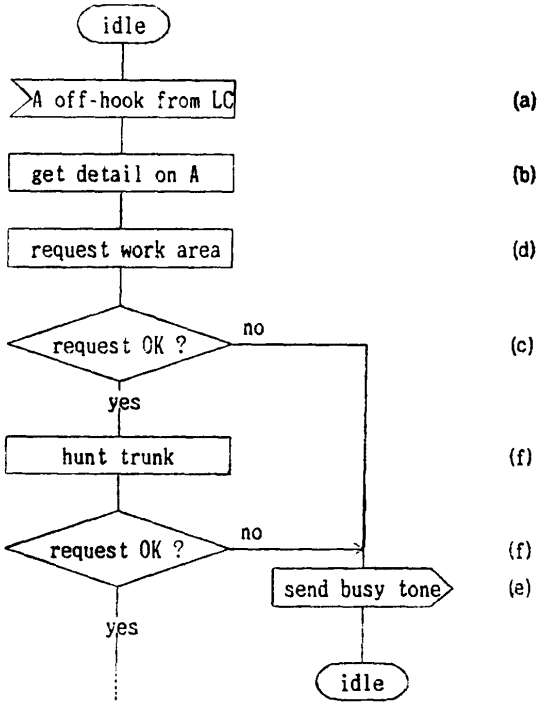


図 8 出力された D-SDL
Fig. 8 Detailed SDL to be outputted finally.

構成要素Bに対しても，同様な推論を実行し，出力要素(f)を得る．ただし，出力 send busy tone は既に存在するので，挿入は省略される．

上記の全処理過程を経て，出力された D-SDL を，図8に示す．

5.3 コード生成サブシステム

5.3.1 サブシステム構成概要

本サブシステムは，D-SDL からのソースコード出力機能を持ち，①パーサ（文字テキスト表記の D-SDL をフレームに変換），②インタプリタ（知識ベースを参照して，フレームからプログラムソースコードを生成），③オプティマイザ（生成されたソースコードを最適化する），④知識ベース（データアクセス条件やサブルーチン呼出し条件等の規則を格納）から成る．

5.3.2 コード生成上の特徴

D-SDL で記述された呼処理ソフトウェアの動作概要は，図9に示すように，ソースコードの一連のデータ操作あるいはサブルーチン呼出しにほぼ1対1で対応付けられる．

この特質を踏まえた上で，コード生成処理上の特徴を以下に示す．

(1) 文脈依存性への対応

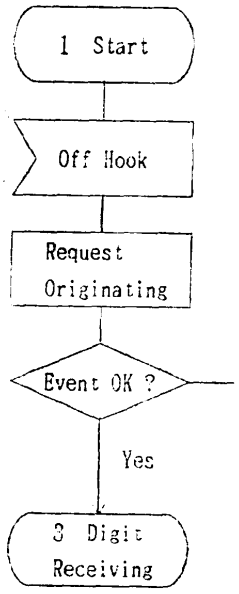
図9の "Event OK?" という D-SDL 記述は，直前の処理で要求された "Originating" を判断するという意味であり，評価の対象となる変数は直前の処理に関連付けられて決定されるため，文脈に依存した D-SDL の解釈が必要となる．本サブシステムでは，解釈ルール関数の導入により，文脈依存パーサを実現する．

(2) アクセスパスの省略への対応

呼制御テーブル (CCB) のような複雑な構造を持つデータへのアクセスに関しては，その具体的なアクセス方法までは D-SDL で記述されないため，これを補ってコード生成を行う必要がある．本サブシステムでは，フレーム化された D-SDL を解釈する際，知識ベース化されたデータアクセス条件を検索しながらデータアクセスパスを補うことにより，ソースコードの生成を行っている．

(3) 生成コードの最適化

交換機の処理性能に直接影響を及ぼす呼処理ソフトウェアでは，生成するコードの最適化が必要である．本サブシステムでは，出力するソー




```
CASE (FCST.STN#R9) OF 0:3
  COND 1
    FCST1.WORK#R9 -> R5
    CASE (SSMPRC.SSMEV#R5) OF
      COND SSMEV.OFFHOOK
        SSMORGR
        (FCST1.WORK#R9 -> R5)
        IF (DTRD.EVENT#R5)=0
          THEN
            SSMNEXT 0,3
        END
    END
END
```

(I) D-SDL (II) コード

図 9 D-SDL とコードの対応
Fig. 9 Correspondence between the detailed SDL and the code.

スコードの記述言語に応じて最適なコード生成が行えるよう，ソースコード言語ごとに最適化処理部を設ける構成としている．

5.4 SDL 検証サブシステム

本サブシステムは，仕様の妥当性・正当性の検証を支援するため，SDL レベルで擬似実行する，いわゆるラピッドプロトタイピングシステムである．基本的には，SDL/GR をシンボルごとに仮想的に実行するが，複数プロセス間の信号の送受信において，システム側で同期をとることを特徴とする．検証のための基本的なコマンド機能としては，シンボルごとのステップ実行機能（前進および後退）を持つ．その他，拡張検証項目を次に示す．

(i) プロセス間の送受信（信号送受信の正しさ，dead-lock の回避，受信信号の処理抜け）
(ii) データの設定，参照（データアクセス論理）
(iii) 信号の後処理（信号送受信後の一定処理）
(iv) 状態間の不可逆性（状態の遷移順序が自然法則に合致しているか）
(v) 状態間のタイミング（規定時間内の処理完了）
(vi) 広域最適化（冗長ステート/冗長タスクの除去，重複部の省略）

6. むすび

本論文では，交換ソフトウェアの呼処理プログラムの自動生成に，段階的な知識処理技術を導入し，知識ベースの構築と推論方式の実現性について示した．

今回試作したプロトタイプシステムは，UtiLisp で記述した結果，プログラムの規模は，グラフィックルーチンを含んだ総行数で，約 20 K ステップとなった．規則数は，タスク生成の場合，G-SDL 要素知識が 5 種類，D-SDL 要素知識が 36 種類であり，コード生成の場合には，生成規則が 145 種類，データ構造の規則が 23 種類である．G-SDL として 15 個の SDL/GR 記号をタスク生成システムに入力した結果，D-SDL として 62 個の SDL/GR 記号が出力された．FACOM M-180 上で，タスク生成に要した CPU 時間は 60 秒である．この出力を，コード生成システムに入力した結果，ソースコードが 200 行生成され，その実行に要した CPU 時間は 2 秒である．

本システムの開発結果から明らかにされた，知識処理に伴う技術的課題を次に示す．

① 既存ソフト資産との品質保証（実績のある既存ソフト資産と新規に生成されたソフトとの総合的な品質の確認）
② 知識獲得（設計者からの知識収集と知識ベースの構築方法）
③ 仕様と知識との非同期性（新規のサービス仕様に関する知識ベースの早期確立対策）
④ 部分的自動プログラミング（一括生成でなく，部分的なプログラム生成のための機能：差分仕様からの部分生成）

今後は，以上の点からも検討を継続して進めて行く予定である．

謝辞 最後に本研究に当たり，多大の御支援を頂いた情報処理研究部門上原部長，杉本主管研究員，人工知能研究部員吉田裕之氏，ならびに通信事業推進本部中村部長，関係各位に対し，謝意を表します．

参考文献

1) Winston, P. H.: *Artificial Intelligence*, Addison-Wesley, Mass. (1977).
2) CCITT Recommendations Z. 100 シリーズ．
3) 鈴木，藤本，広瀬，加藤，吉田：SDL（通信ソフトウェア機能仕様記述言語）支援システム，第 28

回情報処理学会全国大会論文集, pp. 561-562 (1984).
4) 加藤, 吉山, 広瀬, 鈴木：知識ベースを用いた SDL 支援システム, 情報処理学会研究会資料, 知識工学と人工知能, 34-5, p. 8 (1984).
5) Fujimoto, H., Suzuki, T., Katoh, H. and Yoshida, H.: Telecommunications Software Design Support System Based on Specification Languages, *EUROCON '84*, pp. 271-275 (1984).
6) Nakamura, K., Fujimoto, H., Suzuki, T., Itoh, Y. and Katoh, H.: Telecommunications Software Development Environment Based on SDL, 2nd SDL Forum (1985).

(昭和 62 年 4 月 9 日受付)
(昭和 63 年 2 月 10 日採録)

藤本　洋（正会員）

昭和 16 年生. 昭和 40 年日本大学理工学部電気工学科卒業. 同年より東京大学生産技術研究所に勤務. 昭和 44 年富士通(株)に入社. 以来, 通信システム（特に電子交換システム）のソフトウェア開発に従事. 現在, ヒューマンベースのソフトウェア生産技術について興味を持つ. 電子情報通信学会, AAAI 各会員.

鈴木　忠道（正会員）

昭和 25 年生. 昭和 48 年東北大学工学部電子工学科卒業. 同年富士通(株)に入社. 以来, 通信ソフトウェアの開発支援技術に関する研究開発に従事. 特に, ソフトウェア設計作業における思考支援に興味を持つ. 電子情報通信学会会員.

加藤　英樹（正会員）

昭和 28 年生. 昭和 52 年東京工業大学工学部電子物理工学科卒業. 昭和 55 年同大学大学院情報工学専攻修士課程修了. 同大学工学部助手を経て, 昭和 57 年(株)富士通研究所入社. 以来人工知能関係の研究開発に従事. 現在ニューロコンピュータの研究開発を行っている. 人工知能, 人間に興味を持つ. 電子情報通信学会会員.