経済学研究 47-1
北海道大学 1997.6

生産スケジューリング事例ベース構築に関する問題定義／数理モデルの記述形式の研究

鮑　金源

1．はじめに

組織における意思決定の成功にはモデルの利用がかなり重要である（Bharadwaj and Choobineh 1992)。それゆえ，意思決定支援システム（以下DSS）での情報資源の利用は，データの蓄積と解析の重視から信頼性の増加に変わりつつある。モデル管理（model management）の研究が次第に発展し，多数の実験的，商業的モデル管理システムが開発された(Blanning 1993)。

DSSの研究は意思決定者を支援することを目指し，意思決定問題を反映しそれを解決することを取り扱っている。意思決定は本来，その品質が実際問題への認識度合に関わる。意思決定問題の数理モデルと実際問題との整合性がその認識度合に左右される。意思決定の過程では，数理モデルを構成する要因を試行錯誤的に変更してテストすることが不可欠なので，数理モデルの元になる問題定義を繰り返して参照できることが重要である。特に数理モデルの構築や変形の過程で，数理モデルにおける構成要素の関係を決定する問題要因への参照が欠かせない。したがって，モデル管理には従来のように数理モデル化から後の過程を支援するだけでは不十分であり，問題定義を構造化して記述し，問題定義と数理モデルの対応関係を明らかにすることが必要と考えられる。

近年，実用化を強く意識したスケジューリング技術が盛んになりつつあり，産業界では①環境の変化に迅速に対応できる，②現場の制約条件を反映できる，③現場が容易に使える，④階層化され柔軟性がある，という性質を有するスケジューリング技術が待望されている（松本1994)。しかし，従来のスケジューリングに関する研究は理論が重要視され，実際への応用が不十分であるとされる。これを改善するために，MacCarthy and Liu（1993）は，①伝統的なスケジューリング理論をもっと応用指向に位置づける，②伝統的なスケジューリング理論に基づいてその成果を進化させ，豊富なメソッド・技術を開発する，③伝統的なスケジューリング理論におけるアルゴリズムと成果の合併による強力なソフトウェアを開発する，ことが望ましいと指摘している。実際問題と数理モデルの関係を明示的に扱うことは，理論研究の応用を促進する効果が期待できる（関口1994)。こうした理由で，モデル管理支援システムの構築が生産スケジューリングにとって特に意義がある。

しかし，スケジューリング問題は，それを決定する要因が，多くの場合，モデルのなかに直接反映されない。問題の種類がその定義環境に大きく左右される。Nagar他（1995a）は問題の性質，ショップ形態，応用された問題の解法，評価基準などの面から問題を分類した。スケジューリング問題の特性，したがって，それを利用する解法は往々にして，モデル型の定義を少し変更しただけで大きく変化する（関口1996)。モデル管理において，モデル型の決定要因を変更するには，詳細なる実際問題の物理的意味を反映する問題定義への操作が不可欠である。生産スケジューリングにおけるモデル管理支援シス

テムの構築のためには，その前提となる構造化した問題定義の記述形式を明らかにし，問題定義と数理モデルの対応関係を明示的に扱えるようにすべきである。

本研究では，モデリング・ライフサイクル（modeling life cycle, Bharadwaj and Choobineh 1992）の全般的支援の前提となる問題定義と数理モデル構築の支援に注目して，問題定義／数理モデルの記述形式を検討し，生産スケジューリング事例ベース構築の基本方針を明らかにする。本稿では，2節でモデル記述の研究状況を概観し，3節では事例ベースによるモデル管理支援における問題定義／数理モデル記述の作業を検討する。これによって明らかにされる問題定義／数理モデルの記述形式に対する要求を考慮して，4では実体－関連概念を拡張した問題定義／数理モデル及び両者の関連付けの記述形式を定義する。5節では4節で行った定義が3節で検討した要求を満足することを，主に例題を利用して示すとともに，この定義の特徴を検討する。

2．問題定義／数理モデル記述の研究状況

2．1　モデリング・ライフサイクルにおける問題定義とモデル化

モデル管理はモデリング・ライフサイクルの各段階の活動を対象にして研究されている。次の活動がモデリング・ライフサイクルに含まれる。

- 問題定義（problem definition）：意思決定問題の目的，領域，構成を明らかにする。
- モデル化（formulation）：意思決定問題を数理的形式で正確に表現する。
- 解法開発（solver development）あるいは解法選択（solver selection）：意思決定問題を解くのに適切な解法を選択する。そのような解法が見当たらない場合にはそれを開発する。
- 問題解決（solution）：データを集積して，選択か開発された解法を用いて意思決定問題を解決する。
- モデル解釈（interpretation）：仮説の検証，感度分析などを通して，解答をテストし実際問題に即して解釈する。
- モデルの評価（evaluation）と改善（modification）：モデルの信頼性を明らかにし，モデルを改善する。

本研究ではこのようなモデリング・ライフサイクルの全般活動のうち，問題定義とモデル化を取り上げる。

(1)問題定義

問題定義はある規則にしたがって，意思決定問題をその目的とその中に含まれる意思決定要因によって形式的に表現し記述する。

ORプロジェクトにおける意思決定問題の解決前提となる問題定義は，意思決定者，意思決定の実施担当者，OR研究者，問題に関わる他の関係者など，幅広い範囲の関係者が参加することが必須で，しかも，複雑かつ知的な作業であり高度な創造性が要求される。

問題定義の段階で意思決定者は組織における目標の達成のために，問題を提起する。意思決定の実施担当者は問題定義に当たって，経営の直接的な知識に基づいて，目的を達成するために解決すべき課題，利用可能な資源などの制約条件を明確化する。また，計画部門，会計部門，生産部門などの関係者が，各自の立場から問題を洞察して問題定義に関与する。

問題定義の過程では，認識のミスや誤解，データの不足などが絶対に無いとは言えない。それに対して，関係者の間のコミュニケーションを通して，ミスや誤解の訂正，データの補足などが常に必要である。つまり，適切かつ正確な問題定義には，繰り返しと発見的作業が不可欠である。この段階では，問題の定義領域，定義領域に含まれる決定要因，決定要因間の論理関係などの解明が主な作業である。コンピュータ基盤のモデル管理環境では，モデリング・ライフ

サイクルの一環である問題定義の結果を構造化して記録すれば，それを根拠として，モデル構築，モデル変形，解法開発や選択，モデルの意味的解釈などが系統的に統合できるようになろう。

(2)モデル化

「モデル」という用語は研究の立場によって異なる意味で用いられる（Chang他 1993)。ここでは単にモデルと言うときは，数理的形式で表現されたモデル，すなわち，**数理モデル**を指すものとする。意思決定問題を代数的に表現する場合，それを数式モデルとも呼ぶ。

数理モデルの構築は問題の物理的な意味に基づいて問題の抽象化や単純化を施し，問題を数学的形式に反映させる。問題の解決は，既存の数理的解法を選択するにしろ，新しい解法を開発するにしろ，いずれも数理モデルを介して行う。

問題定義に基づいて，意思決定問題を数学的表現にし，数理モデルを構築する。意思決定の実施の効果は，数理モデルが現実状況を的確に表現しているかどうかに大きく依存する（Bharadwaj and Choobineh 1992)。これは二つの面を意味している。一つは問題定義が意思決定問題を正確に記述しているかどうかであり，もう一つは問題定義が数理モデルに適切に抽象化されたかどうかである。数理モデルの的確さは物理的世界の記述である問題定義をどのように抽象化し，単純化するかに依存する。品質の高い数理モデルの構築には，数理モデルの元になる物理的な問題状況を整理した問題定義を参照したり，それを修正したりすることは避けられない。この意味で，意思決定問題の詳細なる決定要因を記述できる問題定義の記述形式が必要となる。

数理モデルの構築では，1つの問題定義に対して複数の数理モデルを構築する可能性がある。例えば，生産スケジューリングにおける順序によって変わる段取り時間問題（Conway他 1967）に対しては次の2つの数理モデルを構築できる。

数理モデル(1)

目的関数：$\min_{[\sigma]} F_{[n]}$

制約条件：$F_{[1]} = s_{[0][1]} + p_{[1]}$

$F_{[2]} = F_{[1]} + s_{[1][2]} p_{[2]}$

……

$F_{[n]} = F_{[n-1]} + s_{[n-1][n]} + p_{[n]}$

数理モデル(2)

目的関数：$\min_{[\sigma]} \Sigma_{i=1,\ldots,n} \Sigma_{k=1,\ldots,n} x_{[i][k]} s_{[i][k]}$

制約条件：$\Sigma_{i=1,\ldots,n} x_{[i][k]} = 1 \qquad k=1,\ldots,n$

$\Sigma_{k=1,\ldots,n} x_{[i][k]} = 1 \qquad i=1,\ldots,n$

$\Sigma_{i\in S} \Sigma_{k\in S} x_{[i][k]} \leqq |S|-1 \quad S \subset \{1,\ldots,n\}$

$x_{[i][k]} = 0$または$1 \qquad i=1,\ldots,n,$

$\qquad\qquad\qquad\qquad\qquad k=1,\ldots,n$

F，s，pはそれぞれ非負実数，σは1,...,nの順列，iとkは順列における要素，Sは {1,...,n} の部分集合である。

逆に，1つの数理モデルが複数の問題定義に対応する可能性もある。例えば，数理モデル(2)は上述の問題定義に対応しているし，巡回セールスマン問題定義にも対応できる。一方，ORにおける数理モデルの効率的な応用（既存成果の再利用）のため，既存の数理モデルを用いて新しい適用領域を探しそこで意思決定問題を解くとか，それを変形して拡大された論議領域の意思決定問題を解決するとかいうような方式がよく見られる。この場合，数理モデルはしばしばそのもとになる問題定義から分離し，汎用的な純粋数学的オブジェクトとして取り扱われる。例えば，線形計画モデルや動的計画モデルの新しい領域への応用がその例である。したがって，モデル管理の立場では問題定義を数理モデルとは独立に行うのが望ましい。

2．2　モデル記述の先行研究

DSSやモデル管理の先行研究において，モデル記述に関するさまざまなアプローチが提案されている。

Chen（1976）がデータ構造を記述するため

に関係データモデル（Codd 1970）の拡張として実体－関連モデル（Entity-Relationship Model，以下ERM）を提案した後，ERMはたびたび拡張され，モデルの記述に用いられている。Blanning（1986）は関係モデルとERMを援用してモデルの表現と操作を調べた。Geoffrion（1987, 1989）は基本実体（primitive entity），複合実体（compound entity），属性（attribute），関数（function），テスト（test）というような要素を利用して，階層的に構成された（hierarchically organized），分割型の（partitioned），非巡回・属性グラフ（attributed acyclic graph）を構成する方法を提案して，モデルやモデルクラスを表現した。Mler-Merbach（1983, 1990）は関係モデルとERMに基づき，システムズアプローチを援用して①標準数理計画モデルの表現，②情報システムにおける数理計画モデルの統合，③大規模な数理計画モデルの組織的な設計のプロセスなどを調べた。その中で，問題叙述レベル（verbal level），システムレベル（system level），実体集合レベル（entity set level），関数レベル（function level），データレベル（data level）という5段階からなるモデル化方法論を提案した。数理モデルを記述するために，システムレベルで実体型や関連型の各要素を識別するために識別属性としてインデックス（index）を導入し，関数レベルで演算子フィールド（operator fields）を導入して関連する属性を1つの関数ブロック（function block）に連結することで関数を表現した。Choobineh（1991）は数理モデルの記述のために，ERMと実体－関連ダイアグラム（ERD）には，数理計画に共通する実体や関連の属性の間に算術（arithmetic），集計（aggregation），比較などの演算関係のような制約条件の型（type of constraints）を表現する概念とアイコン（icons）を，また，関連を定義する実体型の性質をもっと精密に描き出すために，実体の最小要素数（min-card）と最大要素数（max-card）を導入し，線形計画モデルの記述を試みた。

Liang（1993）はモデル管理において，事例ベース推論アプローチの立場から，数理モデルの記述を中心にして，問題の概念（conceptual），構造（structure），関数（function）の各レベルで，実体グラフ（entity graph），属性分類体系（attribute hierarchy），属性型（attribute type）の類似性を提案して，問題とモデルの事例表現，保存，識別，推論を実現し，数理モデルの構築における経験の利用及び学習によるモデリング知識の改善の可能性を示した。Lee and Kim(1995)はBOT（blocks of terms），インデックス（indices），属性（attributes）を用いて，推論によってモデルの構築と記述法を試みた。Ma（1995）は数理モデルを共通のデータを用いる相互に関連する関数の集合をクラス（class）に定義し，継承メカニズムを通して，既存のクラスから新しいクラスを派生するというオブジェクト指向フレームワークによるモデル記述のアプローチを提案した。

関口（1992）はORプロジェクトにおける意思決定問題の解法開発の支援のためCAMPを開発した。CAMPでは項目－詳細アプローチによって，27個の項目に分けてスケジューリングにおける問題を特徴づけた。

これらの研究は，数理モデルの記述を中心にして行われ，数理モデルそのものの記述以外には，多くともそれに直接関連する範囲の情報だけが記載される。モデル管理の立場からすれば，数理モデル構築の元になるのが問題定義であるから，問題定義には，そうした情報よりはるかに多くの情報が記載されるべきである。

従来のスケジューリング問題の分類と記述方法として，Conway他（1967）は4つ組による記述方法A/B/C/D，Graham他（1979）は3つ組による記述方法$\alpha/\beta/\gamma$を提案した。Aは仕事数，Bは作業ステージ数，Cは技術や管理に関する制約，Dは最適化の基準である。αは機械と仕事の関わり具合を，βは仕事の特性や処理順に関する制約などを，γは目的関数を記

述する。これらの記述形式では一般的に，ある種類の問題が与えられたとき，その種類に対応する特定の型記号を各欄に記述するので，問題の型が簡潔に表される。しかも，問題の特徴を高度に抽象したものであり，組合せスケジューリング理論という一つの分野を作り上げるのに有効であった。しかし，これらの記述方法はスケジューリング問題型の変化に容易に対応できるとは言えない。例えば，生産環境の変化，技術の進歩にしたがって，グループ処理，AGV，FMSなどを追加した複雑な問題の記述は難しい。また，各欄の値の間や，同じフィールドに記述される異なる属性の間の相互関係などは不明である。要するに，問題に関する詳細な型の分析，モデル・ベース（model base）の構築などにとって，これらの方法によって記述された情報は十分とはいえない。

これに対して，関口（1994, 1996）は実際問題を記述するに当たって，従来のERMで捉えられた量的属性ばかりでなく，問題の特徴に質的な影響を与える定性的属性も重要視し，問題の記述のために実体－関連アプローチを拡張して，汎実体－関連モデル（以下GERM）を提案している。多種のスケジューリングの問題記述を試みた上で，GERMの問題記述能力の高さ，分解能の高さを主張している。

ERMとGERMは共に，問題を記述するのに問題の物理的な意味を捉え，問題の構成要素を構造化する。この二種類の問題記述法は問題分析の文脈は自然言語に似ており，記述された問題がわかりやすく，人間の認識に適している。両者の異なる点は問題記述のレベルである。前者を用いて記述される問題はただ１つの問題型であり，しかも量的データの構造を定義するだけなので，比較的大きな問題型を記述する。後者は定性的な属性も記述する立場を取っており，異種の問題を記述できることが特徴である。前者を用いて記述された問題は，後者の特例と考えられる。しかし，後者は前者と競合するのではなく，前者と併用して，広範囲の問題を詳細に記述するために提案されている。

3．モデル管理支援の事例ベースにおける問題定義／数理モデル記述の作業

この節では，問題定義や数理モデル記述の作業を効率的，正確かつ再利用可能なように行うには，どのような操作を利用できることが望ましいかを検討する。

3．1　生産スケジューリングにおける事例ベースの概念

長期間を渡って実証された優れている成果を集積し，人間の記憶を補足するためにそれを再利用することが望ましい。これに関しては事例ベースの構築が盛んに注目されている。事例とは，操作的なレベル（operational level）で特定の事情に繋がる知識を表現する形式である（Kolodner 1993）。事例ベースの構築によって，作業結果の蓄積，既存結果の再利用を可能にし，事例ベース推論によって問題の解決案を自動的に作成することも可能である。生産スケジューリングにおける問題定義，モデル構築，解法開発などにとって，事例ベースの構築は特に重要である。これによって，不必要な重複作業を避けて，それらの作業の効率性を高めることができる。

問題定義と数理モデルの構築を対象にする立場では，事例ベースの満たすべき要件は次のように考えられる。

- 事例による知識の表現能力：高度な知的創造活動である問題定義や数理モデルの構築によって生まれる知識，つまり問題定義，数理モデルとそれらに関するノウハウなどが事例によって表現できなければならない。
- 知識の学習機能：問題の記述活動にしたがって，問題定義，数理モデル，それらの構築に関するノウハウなどが生まれる。事例ベースはそれらの知識を絶えず学習し，蓄積すべきである。これによって，知識の増大と

その活用が可能になる。

- 知識の識別機能：事例ベースには，問題定義，数理モデルとそれらの構成要素，ノウハウなどの様々な知識が記憶されている。問題の記述の過程では，知識を効率的に記憶し再利用するために，それらを識別しなければならない。
- 知識の検索機能：問題記述の過程で，再利用の目的で蓄積された知識に対して，必要に応じて，それを検索できなければならない。
- 知識の分解と合成機能：知識の再利用は，知識の異なる表現レベルで行われる。たとえば，事例レベルで再利用し，さらに，事例の構成要素レベルでも再利用するかもしれない。事例ベースはその中に蓄積された知識を再利用する際，必要に応じて異なるレベルで新しい知識を構成したり，あるいは既存の知識を分解するなどの機能を持つべきである。

3．2　事例ベースにおける問題定義／数理モデル記述の要件と操作

事例ベースに記憶する事例を如何に記述するのか，それを如何に操作するかは事例ベースの機能の発揮にとって重要である。次に，4つの視点から検討する。

(1)事例に記述する内容の共通性と相違性

事例ベースの基礎機能である事例の識別や選択は，事例における共通性と相違性に依存する。問題定義／数理モデルの記述は，事例における構成要素の共通性と相違性を反映すべきである。

共通性と相違性による記述内容の分類は，従来，モデルに関して様々な視点から行われ，かつ複雑である。たとえば，線形計画と非線形計画，整数計画，動的計画，待ち行列，ネットワーク，スケジューリングなどの大雑把な問題分類ばかりでなく，線形計画に属する輸送問題や割り当て問題，スケジューリングに属するジョンソン問題やジョブショップ問題などの分類もある。これらの分類では，それぞれのモデルがそれに共通の特徴によって他と区別される。たとえば，線形計画と非線形計画は意思決定変数の次数によって，整数計画は解の値のドメインによって，単一目的関数計画と多目的関数計画はその目的関数の個数によって，待ち行列，スケジューリング，ゲームなどはそれぞれの問題の全体的構造によって，スケジューリングの個別問題型であるジョンソン問題はその問題の物理的な意味によって特徴づけられている。これらのモデルの特徴は純粋に数学的意味のものもあるし，物理的意味を帯びるものもある。たとえば，線形計画や非線形計画や整数計画などは純粋に数学的特徴によって区別されているが，スケジューリングにおける機械の台数や仕事の到着時刻，待ち行列における顧客到着の間隔の分布やサービス場所数などは物理的な特徴と考えられる。この意味で，モデルは数学的な意味だけでなく，問題の物理的な意味によっても判別すべきである。前者の場合，モデルは厳密な数学形式で記述され，単に数学的な関連からモデル型を識別できるので，前述のモデル管理の研究の多くが，それを対象にしてモデルの記述，識別，類似判別などを検討した。後者の場合には，それほど簡潔かつ厳密的な数学形式を持たないか，あるいは多様な数学的な表現が可能なためであろうか，それを対象とするモデルの記述，モデルの識別などに関する検討は意外に少ない。

モデルはその構成要素からなる。モデル自身も構成要素の一つと見なす。モデルの特徴とその構成要素間には，モデルの個々の特徴（以下，型特徴）がその構成要素の特性あるいは構成要素間の関係の特性を表すという関係がある。モデルの構成要素は一般的にある規則にしたがって構造化され，モデルの異なるレベルに分布しているので，型特徴もそれに応じてモデルの異なるレベルに分布しているはずである。異なるレベルの型特徴はその影響範囲（scope）が違

う。すなわち，型特徴はそれを記述するモデルの特定の部分のみに影響する。たとえば，「線形計画」や「スケジューリング」はモデル型の型特徴として見なせる。そのスコープは対応するモデルの全体に及ぶ。決定変数の整数制約は整数線形計画問題の型特徴のひとつと見なせるが，そのスコープは線形計画モデルのなかの意思決定変数にしかない。したがって，モデル識別の実際はモデルの異なるレベルにおける型特徴によって行われる。

Banerjee and Basu (1993) はORの数学的なモデル型を「ルールと（あるいは）プロパティの集合によって特徴づけたモデル具体例の集まり (collection)」と解釈し，かつ環境レベル (environmental level)，構造レベル (structural level)，具体例レベル (instance level)，解レベル (solver level) の4階層に数理モデルの型特徴を捉えて，トップダウンの形式で，モデル型を識別したり選択したりすることを提案した。関口 (1996) は数理モデル作成の元になる問題定義と数理モデルの独立性を重視する観点から，「1つのモデルの数値例の一定の集合」を数学的モデル型と定義し，現実世界の物理的な意味で「ある問題定義の具体例の許容集合」を問題型と定義し，両方の関連付けは結合条件によって実現するアプローチを提案した。

(2)問題定義記述の要件

事例ベースを基盤として問題を定義する場合，問題定義が事例の内容になる。したがって，問題定義の記述は事例ベースの構築にふさわしくなければならない。モデル管理の立場から問題解決の過程と本研究の目的を考慮すれば，問題定義の記述は次の要件を満たすことが必要であると思われる。

- 記述能力：ORプロジェクトによる意思決定問題は幅広い範囲（線形計画，動的計画，待ち行列，ネットワーク，スケジューリング，…）にわたる。同じ分野の問題にしても，問題やモデルの種類が必ずしも同じとは言えない。スケジューリング問題はその例である。問題定義の記述は異種の問題に対応しなければならない。
- 意思決定関係者間のコミュニケーションの可能性：意思決定問題の解決に関与する異なる知的背景を持つ人々の問題と技術に関する相互の情報交換を可能にするために，異なる立場から要求される情報を記述すべきである。
- 問題記述の完全性：問題定義に含まれる情報は，必ずしも数理モデルに直接的に反映することはない。しかし，問題定義として，意思決定のために必要な実際問題の情報を忠実に反映しなければならないし，数理モデルに間接的に反映する情報も記述する必要がある。問題定義としては，必要な情報を漏れなく記述すべきである。
- 一貫性：問題定義では，同一の記述内容に対する操作が異なる時間に跨ったり，複数の作業者によって行われることがある。問題定義の記述は記述の内容やスタイルが問題定義の時間の前後や作業者の違いによって変化すべきでなく，記述された内容の一貫性を維持する機能が重要である。
- 簡易性：問題定義は実際問題から複雑な意思決定要因を抽出して問題定義の構成要素として記述する。抽出された構成要素は，同一の問題定義において何かの関係で相互に関連する。問題定義の記述ではこの複雑な問題定義の構成要素を系統的，かつ簡潔に表現することが必要である。
- 再利用性：問題を定義する場合，同じ意思決定要因が何度も用いられることが多い。たとえば，生産スケジューリングの分野で，仕事や機械などがほとんどすべての問題の定義に使われる。こうした状況に対して，共通の記述された要素を再利用できることが望ましい。
- 作業の効率性：再利用性が作業の効率性を果たす一方で，記述された問題に含まれて

いる問題定義のノウハウ，要素の相互関係，要素の意味的な解釈を参照できれば，作業の効率性も高まる。このような情報を記述できる記述法が重要である。

現実世界での意思決定問題には，大規模な問題定義が避けられない。また事例ベースによる問題定義では，共通性による既存の類似事例の再利用によって作業の効率性を図る。類似事例の再利用には，事例に含まれる問題定義の構成要素の分離，抽出，合成などの操作が必要である。大規模な問題定義やこれらの操作に対応するには，モジュール化し構造化した問題定義の記述形式が望ましい。

(3)数理モデル記述の要件

問題記述の一環である数理モデルの構築は，問題定義と数理モデルの相互の独立性及びモデル管理の一貫性の立場からすれば，その記述が問題定義に対応すべきであるし，数理モデルの活用に便利でなければならない。したがって，数理モデルの記述は次の条件を満たすべきだと考えられる。

- 問題定義との対応性：問題定義から独立した数理モデルによる意思決定問題の表現は数理モデルと問題定義の対応によって実現する。対応づけられた数理モデルの数学的意味と問題定義の物理的意味との間には，矛盾があってはならない。1つの数理モデルは複数の問題定義に対応する場合，その構成要素が対応する問題定義の構成要素に適応できなければならない。
- 操作性：数理モデルの再利用では，既存の数理モデルを新しい意思決定問題に適用する際，モデルの構成要素の変更，削除，追加などの操作をする必要がある。この要求を満たすために，数理モデルの構造化した記述形式が望まれる。

数理モデルの問題定義との対応性は数理モデルと問題定義のそれぞれの構成要素間の対応によって決められる。数理モデルの構成要素は2種類に分けられる。1つは数理モデルの顕在要素（変数，パラメータ，定数）で，もう1つは潜在要素（関数の構成関係や制約条件）である。数理モデルの顕在要素は問題定義に明示しなければならない。潜在要素に対しては，モデルの構築や変形などにおいて必要な判断に用いるために，問題定義の中で参考になる情報を記述することが望ましい。

(4)事例ベースにおける操作

事例ベースでの問題定義／モデル構築は，事例ベースに記憶する事例とその構成要素の操作および事例ベースに対する操作によって行う。

事例とその構成要素に関する操作は作成，定義，複製，修正，削除が考えられる。

- 作成は，問題記述のために事例ベースに使われる各種の記述形式のテンプレートを構築する。テンプレートの作成方式としては新規，あるいは再利用によることがある。新規は新しいテンプレートを構築して定義する。再利用は事例ベースに既存の再利用可能な構成要素を組み込んで，それを修正して利用するか，あるいはそのまま利用する。
- 定義は，作成されたテンプレートに対してその具体的な構成要素の内容を定義する。
- 複製は，選択された事例か，事例の構成要素をコピーする。
- 修正は，事例ベースに記憶されている事例の内容を変更するか，複製された事例か，構成要素を変更して新しいものに変換する。
- 削除は，指定された事例や構成要素を事例データ・ベースから取消して，システムの記憶メモリを開放する。

事例ベースに関する操作は事例の登録，検索，照会，ドロップ，変更を含む。

- 登録は，問題記述に定義された事例を事例ベースに登録する。
- 検索は，指定された情報に基づいて対応する事例ベースから検索情報を満たす事例を

表1　GERMによる問題記述の拡張要素

概　念	意　　味
問題型	ある問題定義の具体例の許容集合
システム型	相互に関連するオブジェクト型の組み合わせとして定義され，上位レベルの実体型
構成型	システム型を構成するオブジェクト型
型属性	オブジェクトの「型」特徴を記述する属性
汎型	型属性を付けているオブジェクト
汎システム型	システム型の各構成型をその汎型に置き換えて作られるシステムの集合
具体例属性	従来のオブジェクトの属性
モデル型	1つのモデルの数値例の一定の集合で、数学的オブジェクト
応用モデル型	問題領域の物理的意味を帯びたモデル型

見出す。

- 照会は，事例ベースの内容をリストにして確認する。
- ドロップは，指定された事例を事例ベースから外す。
- 変更は，選択された事例を別の事例で置き換える。

問題の記述過程では，事例や構成要素に関する操作と，事例ベースに関する操作が密接な連係を保って行われる。

4．実体－関連概念を拡張した問題定義の形式化及び数理モデルとの関連付け

以上の検討から，事例ベースの構築にあたって，問題定義と数理モデルを独立に記述する立場を取り，モデル管理支援のための問題定義／モデル記述の形式化のために，GERMを援用する。しかし関口（1994，1996）はGERMを事例ベース・システムにインプリメントするのに必要な記述形式の厳密な定義を行っていない。そこで本節ではまず簡単にGERMを紹介し（4.1），その事例ベースへのインプリメントに適した記述形式を定義し（4.2，4.3），さらに，数理モデル記述へのGERMの応用を提案（4.4）する。

4．1　汎実体－関連モデルによる問題定義

GERMはERMの拡張として提案されている。ERMにおいては問題の論議領域に取り込まれるモノを実体（entity）として定義する。実体間の対応関係を論議する場合，それを関連（relationship）として定義する。生産スケジューリングの場合であれば，仕事（job）や機械（machine）などが実体でありうる。作業などが仕事と機械の対応関係である関連を表す。問題領域における同種の実体（entity set）や関連（relationship set）を把握するには，実体型（entity type）や関連型（relationship type）を定義する。この時，個々の実体や関連はその属する型の要素の具体例（instance）といわれる。実体型や関連型の要素を記述するには，その属性（attribute）が使われる。具体例では属性の値が固定されている。実体型や関連型の要素は属性値が変われば，異なる具体例に変わる。

GERMによる問題定義には，従来のERMに使われている構成要素に加えて，問題型，システム型，モデル型，型属性，汎型，具体例属性などの概念が使われている（表1）。また，GERMでは問題型と応用モデル型を区別している。問題型と応用モデル型とは同一問題の二つの側面である。つまり，問題型は実世界から抽象されたもので，応用モデル型は数理モデルに物理的意味を付けたものである。属性は属性値の作成方式に基づいて所与属性，派生属性に区別されている。

4．2　汎実体－関連モデルによる問題定義の形式設計

GERMによって表現される情報とその相互関係を3節における検討に照らし，記号

：＝	構成
{...}	繰り返し項目
... ＋ ...	接続
... | ...	選択
[...]	ブロック
(...)	自由項目

を利用して，問題定義のための記述形式を次のように定義する。

システム型の記述形式：

システム型：＝ 名前＋定義説明＋{具体例型属性}＋{構成型}
構成型：＝ [実体型｜関連型｜システム型]
実体型：＝ 汎実体型名＋{具体例型属性}
関連型：＝ 汎関連型名＋{定義型}＋{具体例型属性}
定義型：＝ [実体型｜関連型｜システム型]
具体例型属性：＝ 型属性名＋[型属性値番号｜{型属性名}]

汎システム型の記述形式：

汎システム型：＝ 名前＋定義説明＋{型属性}＋{汎構成型}
汎構成型：＝ [汎実体型｜汎関連型｜汎システム型]
汎実体型：＝ 名前＋定義説明＋{型属性}
汎関連型：＝ 名前＋定義説明＋{汎定義型}＋{型属性}
汎定義型：＝ [汎実体型｜汎関連型｜汎システム型]
型属性：＝ 名前＋定義説明＋[型属性ドメイン｜{型属性}]
型属性ドメイン：＝ {型属性値番号＋[定性的特性値｜数学的特性値｜評価基準値]}
定性的特性値：＝ {定性特徴＋定義説明}
数学的特性値：＝ (具体値)＋ステータス＋(定性的特性値)＋内実＋(テスト条件)
評価基準値：＝ 評価基準＋{型属性}＋(補充情報)＋定義説明
内実：＝ データ型＋(次元)＋(単位)＋(補充情報)
名前：＝ 文字列型
定義説明：＝ 文字列型
具体値：＝ 文字列型
ステータス：＝ 文字列型
テスト条件：＝ {文字列型}
評価基準：＝ 文字列型
補充情報：＝ {文字列型}
データ型：＝ 文字列型
次元：＝ 文字列型
単位：＝ 文字列型
型属性値番号：＝ 文字列型
定性特徴：＝ 文字列型

上の定義の考え方は以下に説明する。

(1)型属性と具体例型属性

GERMでは，異種の問題を記述するために，問題の特徴をオブジェクト型（実体型，関連型）に付けた型属性の値，すなわち，具体例型属性として捉えていてる。それによって記述される情報は従来のERMではオブジェクトの属性に取り上げられなかった量的でない特性，オブジェクトの集合に関する特性，及び従来の量的属性の内実（データ型，次元，単位，意味など）などである。オブジェクト型はその具体例型属性の値によって規定される。このような性質の型属性を記述する形式は次のように整理できる。

具体例型属性は，問題型を構成する実体や関連の集合の特徴を示すものであり，具体的には，識別子と値からなる。識別子は個々の型属性を識別するために用いる。具体例型属性は汎型の型属性に値を定めたものであり，この値は型属性のドメインの1つである。**型属性**は名前，定義説明，ドメインからなる。**ドメイン**は型属性が取りうる値の許容範囲である。型属性は定性的性質と数学的性質の2つの側面を持つ。定性的性質は記述対象の量的でない性質を表現し，数理モデルの潜在要素に対応する。数学的性質は記述対象の量的性質を表現し，数理モデルの顕在要素に対応する。従って，型属性の値としてはこの2つの側面の性質を表現すべきである。

型属性の定性的性質を表現する型属性の値を**定性的特性値**（qualitative characteristic value）と呼び，その表現は値を表わす定性特徴と定性特徴に関する定義説明によって実現する。**定性特徴**は問題の定性的性質に関する簡潔な表現語である。定性的特性値は単一の定性特徴から成るとは限らず，幾つかの定性特徴の組合せによって表現される場合もある。たとえば，スケジューリング問題における「仕事」が汎実体型として定義されるとき，「仕事」が「処理条件」という型属性を持つ。後述のジョンソン問題（Johnson 1954）の定義においては，「処理条件」は定性的特性値を取り，その値は「優先処理なし」や「分割禁止」のような定性特徴

によって表現される。

型属性の数学的性質を表現する値を**数学的特性値**（mathematical characteristic value）と呼ぶ。数学的特性値は記述対象の属性が数学的オブジェクトになる数学的情報である。記述対象の数学的オブジェクトになる属性は，その属性のステータス（status），内実，テスト条件の情報を持つ。**ステータス**はオブジェクト型のインスタンス属性値の定め方を表し，「変数」，「所与」，「派生」，「定数」などの文字列型定数を取る。**内実**は数学的オブジェクト（変数，パラメータ，定数など）の本質的内容を表し，その中にはデータ型，次元，単位などの情報を含む。これは問題の数理モデルの構築，問題定義と数理モデルの対応づけ，問題と解法の対応づけ，モデルの具体例データのデータベースからの取得などにとって重要である。**テスト条件**は問題の具体例においてその特性値が取りうる値の許容範囲を記述する。たとえば，ジョンソン問題では，「作業」を汎関連型として定義する。「作業」の型属性である「処理時間」は数学的オブジェクトになりうる。その数学的特性値のステータスは「所与」，内実の中のデータ型は「実数」であり，次元や単位は特に指定しない。テスト条件は「正数」である。

1つの型属性によって表現される特性には定性的特性値と数学的特性値の両方を持つ場合がある。たとえば，上述の「処理時間」は，「確定的」という定性的特性値を追加することによって，その特性が一層明確になる。また，型属性はサブ型属性を持つ場合もある。つまり，一般的には型属性は木構造を持つ（鮑・関口 1995）。サブ型属性はそれより上位レベルの型属性（親型属性）をさらに詳細に記述する役割を果たす。

(2)汎型と具体例型

汎型（汎実体型，汎関連型，汎システム型）はそれを記述する型属性を定義することによって定義される。ここで定義する記述形式が現実世界の問題記述の必要に応じて拡張できるためには，汎型の型属性の種類やそのドメインが動的に拡大できなければならない。GERMによる問題型の指定には，そこに含まれる型属性に具体値を指定すると共に，記述しようとする実際問題に含まれる汎型に対応する固定的な実体型や関連型に具体例を指定しなければならない。固定的な実体型や関連型とは問題型に対応するシステム型において，その具体例毎に変化しない構成型である。例えば，ジョンソン問題では機械型は固定型の1つである。汎型のすべての型属性に値が指定されたとき，**具体例型**（実体型，関連型，システム型）と呼ぶ。例えば，ジョンソン問題では実体型である「仕事」の型属性「到着時刻」に具体値「静的」が指定される，などである。

汎システム型は相互に関連する汎型から構成される。それを構成する汎型を**汎構成型**と呼ぶ。同様に，**システム型は**相互に関連する具体例型から構成される集合体である。それを構成する具体例型を**構成型**と呼ぶ。汎システム型とそれから派生されるシステム型は構成が類似である。汎システム型，システム型はそれぞれそれより上位レベルの汎構成型，構成型としても用いられる。汎システム型には型属性を定義でき，その派生型であるシステム型には対応する具体例型属性を含めるので，それらをモジュールである汎構成型や構成型として他の汎システム型，システム型を構成できる。これによって，問題定義の結果を他の問題定義の組み込み型として使えるようになる。汎システム型やシステム型には，特有の属性としてシステム型を評価する情報を記述する評価属性がある。それを型属性に収めると，その値は**評価基準値**によって表現する。評価基準値にはシステム型を評価する属性を指定する情報と，評価基準の情報を含む。

汎型および型属性の情報は対応する具体例型が定義されたとき，必要に応じて生成ないしは追加される。このようにして汎型に保持された情報を使って，新しい具体例型を指定（派生）できる。このメカニズムによって，記述の拡張

表2　ジョンソン問題のシステム型

名前	定義説明	具体例型属性		構成型
		型属性名	型属性値番号	
Johnson	ジョンソン問題（Johnson, 1954）とは，2機械，n個仕事のフローショップで，仕事を処理する要求は最大完了時刻を最小にすることである。この時のスケジュールを求める。	評価属性 最大完了時刻 順序スケジュール	評価属性_値1 最大完了時刻_値1 順序スケジュール_値1	仕事型 機械型 作業型 バッファ型 ルーチング型

表3　ジョンソン問題の実体型である構成型

名前	具体例型属性	
	型属性名	型属性値番号
仕事型	仕事数	仕事型_仕事数_値1
	仕事ＩＤ	仕事型_仕事ID_値1
	到着時刻	仕事型_到着時刻_値1
	処理条件	仕事型_処理条件_値1
	同時処理機械数	仕事型_同時処理機械数_値1
	処理開始時刻	仕事型_処理開始時刻_値1
	処理完了時刻	仕事型_処理完了時刻_値1
機械型	機械数	機械型_機械数_値1
	機械ID	機械型_機械ID_値1
	稼動条件	機械型_稼動条件　値1
	同時処理可能仕事数	機械型_同時処理可能仕事数_値1

表4　ジョンソン問題の関連型である構成型

名前	定義型名	具体的型属性	
		型属性名	型属性値番号
作業型	仕事型	処理条件	作業型_処理条件_値1
	関連型	処理順序	作業型_処理順序_値1
		処理時間	作業型_処理時間_値1
		処理開始時刻	作業型_処理開始時刻_値1
		処理完了時刻	作業型_処理完了時刻_値1
バッファ型	機械型	容量	バッファ型_容量_値1
ルーチング型	作業型	工程順形態	ルーチング型_工程順形態_値1

性を確保すると共に，問題定義の記述の一貫性が保ちやすくなる。

4．3　問題定義例——ジョンソン問題

以上の記述形式を利用して問題を定義するには，具体例型と汎型の定義を通して行う。具体的には次のような分析ステップが必要である。

1. 問題構成の解明：実際問題を表現するシステム型がどんな構成型によって構成されるか，構成型の間にどんな関係を持つかを明らかにする。
2. 型属性の解明：システム型やその構成型を記述するには，それぞれにはどういう型属性が必要であるかを解明する。
3. 型属性値の指定：各々の型属性にはどんな値を指定して問題の特徴を反映するか，具体的に指定する。
4. システム型の確立：問題定義に対して，問題定義を記述するシステム型を定める。
5. 汎型の修正：汎型には，それによって派

生されたあらゆる具体例型の情報を保持しているので，新しく定義した具体例型における型属性や型属性値の情報が対応する汎型に保持していなければ，あるいは既存の情報が修正された場合，それらの情報を対応する汎型に反映する。

次にジョンソン問題を例にして，問題定義の記述を示す。

(1)具体例型の定義

ジョンソン問題を定義するのに，対応するシステム型を「Johnson」とする。ジョンソン問題の分析によって，その中に含まれる仕事と機械は実体型の「仕事型」と「機械型」で，機械で仕事を処理する作業は「仕事型」と「機械型」を定義型とする関連型の「作業型」で，機械と機械の間のバッファは「機械型」を定義型とする関連型の「バッファ型」で表現できる。仕事を処理する工程順は一般的に仕事と機械の両方によって決定されるが，その両方の関係はすでに「作業型」によって表現されているので，「作業型」を定義型とする関連型の「ルーチング型」によって表現できる。したがって，「Johnson」は「仕事型」，「機械型」，「作業型」，「バッファ型」，「ルーチング型」によって構成される。これらの具体例型の具体例型属性を検討して，表2，3，4のように記述した。

「Johnson」を評価するには，具体例型属性の「評価属性」を定義する。その評価指標は「最大完了時刻」，それを決定するのは「順序スケジュール」である。したがって，「Johnson」の具体例型属性として「順序スケジュール」と仕事の「最大完了時刻」も定義すべきである。これに関する記述は表2に反映している。

具体例型属性の値は，それが表現する問題定義情報がシステム型の評価属性であるか，数学的であるか，定性的であるかによって定まる。ジョンソン問題定義に使われる具体例型属性を分析すれば，次のことがわかる。システム型の「評価属性」が「最大完了時刻」を「最小」にする評価基準値を持つ。システム型を定義する最大完了時刻・順序スケジュール，「仕事型」を定義する仕事数・仕事ID・処理開始時刻，「機械型」を定義する機械数・機械ID，「作業型」を定義する処理順序・処理時間・処理開始時刻・処理完了時刻が数学的特性値を持つ。「仕事型」を定義する到着時刻・処理条件・同時処理機械数，「機械型」を定義する稼動条件・同時処理可能仕事数，「バッファ型」を定義する容量，「ルーチング型」を定義する工程順形態が定性的特性値を持つ。また，「作業型」を定義する処理時間の値には定性的特性値が含まれる。ジョンソン問題定義の具体例型属性値は表5，6，7に示した。

以上の分析によって，各々の具体例型の内容が明確にされ，ジョンソン問題を表現するシステム型の「Johnson」も確立できた。

(2)汎型の定義

ジョンソン問題の定義が終われば，その中に記述された情報は，それぞれの対応する汎型に反映すべきである。「Johnson」の対応する汎システム型を「2M-F」とする。ジョンソン問題定義の汎型に反映する形式記述は表8，9，10，11に示した。

4．4　数理モデルの記述形式及び問題定義との対応づけ

数理モデルは，目的関数，制約条件，定数，係数，変数のような構成要素からなる。これらの構成要素の表現には，モデリング言語として多数の形式が提案されている。たとえば，Fourer他（1990）はモデリング言語（AMPL）の構築において，数理モデルを集合，パラメータ，変数，目的関数，制約条件式という5つの主要部分から捉えている。ここではGERMアプローチを援用する数理モデルの記述形式を提案する。その後，問題定義と数理モデルの対応関係を記述する形式をやはりGERMアプローチを援用して提案する。

表5　ジョンソン問題における評価基準値タイプの型属性値

型属性値番号	評価基準	型属性名	補充情報	定義説明
評価属性_値1	最小	最大完了時刻		仕事の最大完了時刻を最小にする。

表6　ジョンソン問題における数学的特性値

型属性値番号	具体値	ステータス	定性的特性値	内実				テスト条件
				データ型	次元	単位	補充情報	
最大完了時刻_値1		派生		実数				
順序スケジュール_値1		変数		順列	n			
仕事型_仕事数_値1	n	所与		整数				n>0
仕事型_仕事ID_値1		ID		整数				0<ID<=n
仕事型_処理開始時刻_値1		派生		実数				0以上
仕事型_処理完了時刻_値1		派生		実数				0以上
機械型_機械数_値1	2	所与		整数				
機械型_機械ID_値1		ID		整数				0<ID<=2
作業型_処理順序_値1		変数		整数				1,2, ..., n
作業型_処理時間_値1		所与	作業型_処理時間_値1_特徴値	実数				0以上
作業型_処理開始時刻_値1		派生		実数				0以上
作業型_処理完了時刻_値1		派生		実数				0以上

表7　ジョンソン問題における定性的特性値

型属性値番号	定性特徴	定義説明
仕事型_到着時刻_値1	静的	すべての仕事は工場に同じに到着する。しかも工場は遊んでおり，すぐにその仕事に取り掛かれる。
仕事型_処理条件_値1	分割処理禁止	各仕事は分割処理が禁止される。
	優先処理なし	仕事の優先処理はない。
仕事型_同時処理機械数_値1	1	機械の同時に処理する可能な仕事数1である。
機械型_稼動条件_値1	連続	各機械は故障しない。
機械型_同時処理可能仕事数_値1	1	機械の同時に処理する可能な仕事数1である。
作業型_処理条件_値1	優先処理なし	仕事の優先処理はない。
作業型_処理時間_値1_特徴値	確定的	事前に明確である。
バッファ型　容量_値1	無限	中間置き場の許容量は無制限である。
ルーチング型_工程順形態_値1	フローショップ	各仕事の工程順が同じである。

(1)数理モデルの記述形式

数理モデルは記号型と数式型なる2つのオブジェクト型からなるという立場を取って，オブジェクト型の**数理モデル型**によって記述する。

記号型は数理モデルにおける記号と記号によって表現される数学的情報を記述する。記号の数理モデルにおける意味は，それによって表現される数学的情報によって決められる。問題定義

表8　ジョンソン問題の対応する汎システム型

名前	定義説明	型属性	汎構成型
2M-F	仕事，機械によって構成されるショップ型である。これを基準にして，汎システム型や汎構成型などに型属性の登録や，型属性値の変更などによって，多数の問題型を表現できる。	評価属性 最大完了時刻 順序スケジュール	汎仕事型 汎機械型 汎作業型 汎バッファ型 汎ルーチング型

表9　2 M-F の汎実体型である汎構成型

名前	定義説明	型属性
汎仕事型	機械工業での製品，部品などの概念に当てはまる。	仕事数 仕事 ID 到着時刻 処理条件 同時処理機械数 処理開始時刻 処理完了時刻
汎機械型	仕事を処理する設備，入力などである。	機械数 機械 ID 稼動条件 同時処理可能仕事数

表10　2 M-F の汎関連型である汎構成型

名前	定義説明	汎定義型名	型属性
汎作業型	機械で仕事を処理する作業，タスク，工程などに当てはまる。	汎仕事型 汎関連型	処理条件 処理順序 処理時間 処理開始時刻 処理完了時刻
汎バッファ型	2つの処理場の間の貯蔵場所に当てはまる。	汎機械型	容量
汎ルーチング型	各仕事の各機械での処理の工程順である。	汎作業型	工程順形態

表11　2 M-F における型属性

名前	定義説明	型属性ドメイン
2 M-F.評価属性	システムを評価する	評価属性_値 1
2 M-F.最大完了時刻	すべての仕事の処理を完了する時刻	最大完了時刻_値 1
2 M-F.順序スケジュール	個々の機械で仕事を処理する順序	仕事型_仕事数_値 1
2 M-F.仕事型. 仕事数	汎仕事型で定義される仕事数	仕事型_仕事 ID_値 1
2 M-F.仕事型. 仕事 ID	仕事型のインスタンスの識別子	仕事型_仕事 ID_値 1
2 M-F.仕事型. 到着時刻	各仕事の処理の着手可能時刻	仕事型_到着時刻_値 1
2 M-F.仕事型. 処理条件	仕事の処理に関する制約条件	仕事型_処理条件_値 1
2 M-F.仕事型. 同時処理機械数	一つの仕事を同時に処理する機械数	仕事型_同時処理機械数_値 1
2 M-F.仕事型. 処理開始時刻	仕事を処理する開始時刻	仕事型_処理開始時刻_値 1
2 M-F.仕事型. 処理開始時刻	仕事を処理した完了時刻	仕事型_処理開始時刻_値 1
2 M-F.機械型. 機械数	汎機械型で定義される機械の台数	機械型機械数_値 1
2 M-F.機械型. 機械 ID	仕事型のインスタンスの識別数	機械型_機械 ID_値 1
2 M-F.機械型. 稼動条件	各機械の連続的利用能力	機械型_稼動条件_値 1
2 M-F.機械型. 同時処理可能仕事数	機械の同時に処理する可能な仕事数	機械型_同時処理可能仕事数_値 1
2 M-F.作業型. 処理条件	各処理に関する制約条件	作業型_処理条件_値 1
2 M-F.作業型. 処理順序	各仕事の各機械での処理順序	作業型_処理順序_値 1
2 M-F.作業型. 処理時間	各処理にかかる時間	作業型_処理時間_値 1
2 M-F.作業型. 処理開始時刻	処理の開始時刻	作業型_処理開始時刻_値 1
2 M-F.作業型. 処理完了時刻	処理の完了時刻	作業型_処理完了時刻_値 1
2 M-F.バッファ型. 容量	中間置き場の許容量	バッファ型_容量_値 1
2 M-F.ルーチング型. 工程順形態	各仕事の各機械での処理の工程順	ルーチング型_工程順形態_値 1

の記述形式の中で定義される型属性の数学的情報と同じように，その中にはステータス，内実，テスト条件を含んでいる。ステータスと内実は問題定義の記述形式での定義と同じで，テスト条件は記号が数理モデルの中で取りうる値の範囲を示す。こうした数学的情報は対応する問題型に含まれる型属性の数学的特性値に，矛盾のないように対応づけることが必要である。従来の数理モデルの定数，係数，変数という区別は記号のステータスによって識別され，その数値に関する情報は記号の内実によって表現され，数値の妥当性は記号のテスト条件で記述する。

数式型は数式の内容と数式のステータス，テスト条件及びその定義説明を記述する。数式のステータスは「評価関数」,「制約条件」のいずれかの値を取る。テスト条件は数式の妥当性を保証する。数式は記号，演算子によって構成されるが，表記形式としては文字列型の形を取る。

以上の論議に基づいて，数理モデル型の記述形式を次に示す。

表12　ジョンソン問題の対応する数理モデル

名前	定義説明	記号	数式
Mathl	ジョンソン問題定義が応用される数理モデル	n	Min[σ]C
		m	c[s[i]＝1][1]=t[i][1]
		i	c[s[i]][1]=c[s[i]－1][1]+t[i][1]
		j	c[s[i]＝1][j]=c[s[i]＝1][j－1]+t[i][j]
		s[i]	c[s[i]][j]=Max{c[s[i]][j－1],c[s[i]－1][j]}+t[i][j]
		t[i][j]	
		c[s[i]][j]	C=Max[i][j]c[s[i]][j]
		σ	
		C	

表13　Mathl の記号型

記号	ステータス	内実				テスト条件
		データ型	次元	単位	補充情報	
n	所与	整数				n＞0
m	所与	整数			定数	m＝2
i	添字	整数				i=1,2,…,n
j	添字	整数				j=1,2,…,m
s[i]	変数	整数				s[i]=1,2, …,n
t[i][j]	所与	実数				t[i][j]＞0
c[s[i]][j]	派生	実数				c[s[i]][j]＞0
σ	変数	順列	n			σ＝{s[1],s[2],…,s[n]}
C	派生	実数				C＞0

表14　Mathl の数式型

名前	数式ステータス	数式	テスト条件	定義説明
	目的関数	Min[σ]Max[i][j]c[s[i][j]		
	制約条件式	c[s[i]＝1][σ 1 []=t[i][1]		初期条件
	制約条件式	c[s[i]][1]=c[s[i]－1]+t[i][1]	s[i]≠1	
	制約条件式	c[s[i]＝1][j]=c[s[i]＝1][j－1]+t[i][j]	j≠1	
	制約条件式	c[s[i]][j]=Max{c[s[i]][j－1],c[s[i]－1][j]}+t[i][j]	s[i]≠1,j≠1	
	制約条件式	C=Max[i][j]c[s[i]][j]		

数理モデル型：＝ 名前＋定義説明＋{記号型}＋{数式型}
記号型：＝ [記号｜記号[記号]｜…]＋ステータス＋内実＋(テスト条件)
数式型：＝ (名前)＋ステータス＋数式＋(テスト条件)＋(定義説明)
記号：＝ 文字列型
数式：＝ 文字列型

記号型と数式型は特殊な実体型，数理モデル型は特殊なシステム型と見なすことができる。従来のモデル化言語に使われる集合は実体型や関連型に，パラメータや変数や制約条件などは実体型や関連型などの属性や属性間の関係に，目的関数はシステム型の評価属性に相当する。記号型と数式型によって記述された数学的情報は，既存のAMPLなどのようなモデリング言語に使われている数学的情報に対応している。この数理モデルの記述形式を用いて，ジョンソン問題の数理モデルを「Math1」と定義し表12，13，14に示す。

(2)結合条件の記述形式

数理モデルの中に記述された記号や数式は問題定義との対応づけによって具体的な意味を帯びる。問題定義と数理モデルの対応づけとは，問題型の具体例型属性と数理モデル型の記号と

表15　ジョンソン問題定義と数理モデルの結合条件

名前	定義説明	システム型名	数理モデル型名	結合子	
				記号型名	型属性名
Johnson-Mathl	ジョンソン問題とMathl数理モデルの結合条件	Johnson	Mathl	n	仕事.仕事数
				m	機械.機械数
				i	仕事.仕事ID
				j	機械.機械ID
				s[i]	作業.処理順序
				t[i][j]	作業.処理時間
				c[s[i]][j]	作業.処理完了時刻
				σ	順列スケジュール
				C	最大完了時刻

の対応づけを指している。問題定義と数理モデルの独立性に配慮して，この対応づけは次の**結合条件型**によって行う。

結合条件型：＝ 名前＋定義説明＋システム型名＋
　　数理モデル名＋{結合子}
結合子：＝［記号｜数式型名］＋型属性名

数理モデルは問題定義の数学的抽象なので，問題定義の具体例型属性のすべてが数理モデルに現れる訳ではない。したがって，結合条件型に現れる具体例型属性は問題定義にある具体例型属性の一部だけである。ここに現れない具体例型属性は数式の中に間接に反映されている。提案した結合条件の記述形式を使って，ジョンソン問題定義と数理モデルの結合条件を「Johnson__Math1」として表15に示す。

5．提案する問題定義／数理モデルの記述形式と事例ベースの構築

5．1　事例ベースを利用する問題定義／数理モデル記述の作業内容

問題を定義しモデルを構築するには，大量の知識が取り扱われる。事例ベースの構築によって，これらの知識を記憶し活用できるようになる。生産スケジューリング事例ベースでは，4節で提案した記述形式で多数の問題を事例として記憶しているものとしよう。3節で述べた問題記述作業に基づいて，この事例ベースを利用して新しい所与問題を定義する際に必要とされる作業を整理すれば，概ね次のようになる。

- 所与問題の構成解明：所与問題を分析して問題の構成を明らかにする。問題の分析では，その中に含まれる具体例型，具体例型の具体例型属性を解明する。
- 事例検索：事例ベースから類似の既存事例を検索する。これは所与問題の構成型やそこに含まれる具体例型属性を用いれば効率よく実現できる。
- 事例修正：類似の既存事例を所与問題の定義要求にしたがって修正作業を施し，問題を定義する。これは事例ベースから検索した事例の記述がモジュール化していることから，必要なモジュール（オブジェクト型）についてのみ修正作業をすれば良く，効率的であると共に，記述の一貫性を保つのにも都合が良い。
- 事例の評価と更新：新しく定義された問題を評価して事例ベースを更新する。事例ベースの更新は新しく定義された問題事例をオブジェクト型の事例ベースに追加することと，この記述に使われた汎型，型属性あるいは型属性値の中に新しいものが存在するときにはこれを汎型の事例ベースに追加することとからなる。

以上の作業から成る事例ベースによる問題記述の様子を2MF-CF問題（Nagar他 1995b）を例にして示す。ここでは4つの作業が順に記述されるが，これらの作業がこの順に逐次的に進

表16　2 MF-CF 問題のシステム型

名前	定義説明	具体例型属性		構成型
		型属性名	型属性値番号	
2 MF-CF	ジョンソン問題の拡張として，2機械，n個仕事のフローショップで，仕事を処理する最大完了時刻，及び総フロー・タイムを最小にする順序スケジュールを求める具体例である。	……	……	……
		評価属性	評価属性_値2	
		総フロー・タイム	総フロー・タイム_値1	
		総フロー・タイム重み	総フロー・タイム重み_値1	
		最大完了時刻重み	最大完了時刻重み_値1	

表17　2 MF-CF に新しく定義した評価基準値

値番号	評価基準	型属性名	補充情報	定義説明
評価属性_値2	最小	最大完了時刻 総フロー・タイム	多目的評価	最大完了時刻と総フロー・タイムを評価属性として，その和を最小にする

表18　2 MF-CF に新しく定義した数学的特性値

型属性値番号	具体値	ステータス	定性的特性値	内実				テスト条件
				データ型	次元	単位	補充情報	
総フロー・タイム_値1		派生		実数				正数
総フロー・タイム重み_値1		所与		実数		nil		正数
最大完了時刻重み_値1		所与		実数		nil		正数

行するものでは必ずしもない。むしろ，多くのフィードバックを含む試行錯誤的な作業と考えるべきである。

(1)類似事例の検索

2MF-CF問題の分析によって，それをGERMによって定義するには，仕事型，機械型，作業型，バッファ型，ルーチング型の構成型が必要なことが判る。この情報を手掛かりとして，事例ベースに記憶されている同じ構成の汎システム型を検索する。その結果，「2M-F」が検索される。既存問題定義のノウハウを利用するために，「2M-F」に定義されている情報，つまり，各々の型属性，そのドメインに定義されている型属性値などの定義情報を参考にしながら，2MF-CF問題をさらに分析する。結局，「2M-F」に既に定義されている型属性に関する利用可能な情報が明確になる。さらにこの情報を使って，「2M-F」から派生した既存のシステム型を検索して，もっとも類似の事例「Johnson」が検索される。

(2)問題定義の事例修正

検索された「Johnson」に基づいて，2MF-CF問題を定義する。「Johnson」の定義と2MF-CF問題の定義要求を比べて，次の相違点がわかる。

①個々の仕事のフロー・タイム

②問題の評価属性に入るシステムの総フロー・タイム

③問題の多目的評価属性

したがって，類似事例の修正方策として，①は個々の仕事の処理完了時刻に等しいので，2MF-CF問題の定義では特に処理する必要はない，②に対しては2MF-CFのシステム型に，「総フロー・タイム」として定義する，③の場合，評価される具体例型属性の重みをシステム型の具体例型属性に追加定義し，評価属性に多目的評価関数を表す具体例型属性値を定義する。類似事例の修正は次のように施す（修正後の記述はジョンソン問題の記述に対する修正部分だけを示す）。

検索された事例を複製して，その名前を

「2MF-CF」に書き直す。さらにその定義説明にも必要な変更を施す。システム型の具体例型属性として「総フロー・タイム」,「総フロー・タイム重み」,「最大完了時刻重み」を「2MF-CF」に定義する。複製されたシステム型の具体例型属性「評価属性」の型属性値番号をドロップして，多目的評価関数を表す値に修正する(表16)。新しく導入された具体例型属性に対応する型属性値を整理しておく（表17，18)。

(3)数理モデル，結合条件の事例修正

結合条件を通して，類似事例に対応している数理モデを検索する。その結果，「Math1」が検索される。問題定義段階における経験から，それを利用するのが得策とわかるので，複製して名前を「Math2」に書き直す。その中に「2MF-CF」に新規に導入した「総フロー・タイム」,「総フロー・タイム重み」,「最大完了時刻重み」に対応する記号と数式を定義して追加する。「評価属性」の値に対応する目的関数を定義するには，元の内容を多目的関数で入れ替える(表19，20，21)。

新しい結合条件を定義して，「2MF-CF」と「Math2」を対応づける（表22)。

(4)事例の評価と更新

事例の妥当性は意思決定の成功・失敗に決定的な影響を与える。問題定義が実際問題を正確に表現できるかどうか，問題定義の数理モデルによる表現が適切であるかどうかなどの面から新しい事例を評価する。問題定義／数理モデル構築の段階での事例の評価は問題定義者や数理モデル構築者が中心となる。

事例評価を経て，完成した事例を事例ベースに記憶する。事例に含まれるシステム型では対応する汎システム型への変更情報があれば，その情報を汎システム型に反映する。2MF-CF問題の定義には，元の汎システム型「2M-F」に含まれない型属性と新しく定義した型属性値が使われているので，それらの情報を「2M-F」に反映すべきである。反映した結果は表23，24に示す。

5．2　検討

表19　2 MF-CF 問題定義の対応する数理モデル

名前	定義説明	記号	数式
Math 2	2 MF-CF に応用される数理モデル	……	……
		α	Min[σ]{αF+βC}
		β	F=Sum[σ]c[s[i]][j]
		F	$\alpha+\beta=1$

表20　Math 2 に新しく定義した記号型

記号	ステータス	内実				テスト条件
		データ型	次元	単位	補充情報	
α	所与	実数		nil		$\alpha>0$
β	所与	実数		nil		$\beta\leq 0$
F	派生	実数				

表21　Math 2 に新しく定義した数式型

数式ステータス	数式	テスト条件	定義説明
目的関数	Min[σ]{αF+βC}		
制約条件式	$\alpha+\beta=1$		
制約条件式	F=Sum[σ]c[s[i]][j]　F〉0		

表22　2 MF-CF 問題定義と数理モデルとの結合

名前	定義説明	システム型名	数理モデル型名	結合子	
				記号型名	型属性名
2 MF-CF Math 2	2 MF-CF 問題とMath 1 数理モデルの結合条件	2 MF-CF	Math 2	n	仕事.仕事数
				m	機械.機械数
				i	仕事.仕事ID
				j	機械.機械ID
				s[i]	作業.処理順序
				t[i][j]	作業.処理時間
				c[s[i]][j]	作業.処理完了時刻
				σ	順列スケジュール
				α	総フロー・タイム重み
				β	最大完了時刻重み
				C	最大完了時刻
				F	総フロー・タイム

表23　更新された 2 M-F

名前	定義説明	型属性	汎構成型
2 M-F	仕事，機械によって構成されるショップ型である。これを基準にして，汎システム型や汎構成型などに型属性の登録や，型属性値の変更などによって，多数の問題型を表現できる。	……	……
		総フロー・タイム	
		総フロー・タイム重み	
		最大完了時刻重み	

表24　2 MF-F における型属性のドメインの変更

名前	定義説明	型属性ドメイン
2 MF-F.評価属性	システムを評価する	……
		評価属性_値 2
2 MF-F.総フロー・タイム	個々の仕事の完了時刻の総和	総フロー・タイム_値 1
2 MF-F.総フロー・タイム重み	評価属性における総フロー・タイムの重み	総フロー・タイム重み_値 1
2 MF-F.最大完了時刻重み	評価属性における最大完了時刻の重み	最大完了時刻重み_値 1

モデル管理支援のため，GERMを援用して問題定義／数理モデル記述を形式化し，さらに，それを用いた生産スケジューリング事例ベースによるモデル化支援を検討した。ここでは提案した記述形式について検討する。

(1)定性的／数学的型属性値

関口（1994，1996）は型属性の重要性を指摘し，その問題定義における意味を明らかにしているが，型属性の内容を詳しく分析してはいない。本研究では，型属性によって記述する情報の性質によって，定性的型属性値と数学的型属性値に分類した。

定性的型属性値は，数理モデルの構築に数学的オブジェクト間の関係，制約条件の確立のために必要な情報を記載する。数学的型属性値は数学的オブジェクになる属性の数学的情報を記述する。このような型属性をそこに載せた情報の性質によって分離できる記述形式は，モデル管理支援の研究が進めば進むほど，その重要性がますます増える。モデル管理のライフ・サイクルにおける問題型の具体例データの記述にしても，解法開発支援にしても，問題解決にしても，いずれも問題の数学的情報を必要とする。例えば，問題定義／数理モデルの記述形式を問題型やモデル型のレベルで設計したが，実行型のモデル化言語のための具体例データの記述に関しては検討しなかった。しかし，問題定義／

数理モデル記述に含まれる数学的型属性値を用いれば，関係データベースと接続するか，あるいは関係データベースとの併用で具体例データの記述は容易に実現できると思われる。

(2)問題記述と数理モデルの自然な対応づけ

問題定義における数学的特性値の属性として内実を取り入れたことによって数理モデルとの対応が自然になった。

数理モデルでは，変数，定数，パラメータなどの数学的オブジェクトを基本構成要素として扱う。これらの要素の具体的な数学的意味はその内実によって定められる。一方，問題定義では，実際問題を忠実に反映するための検討の対象として，数学的なオブジェクトになる属性の内実が取り上げられるのは自然である。したがって，問題定義と数理モデルを独立に記述する場合，問題定義の記述に数学的特性値の属性として内実が取り込まれ，それを問題定義と数理モデルの結合の橋渡しとして使うことはごく自然である。これによって，問題定義から数理モデルの構築へ，あるいは数理モデルから問題定義へ，容易に橋渡しできるようになる。

(3)結合条件型による問題定義と数理モデルの結合の柔軟性

汎実体—関連アプローチでは，問題記述の構造化やモジュール化を重要視とし，問題定義と数理モデルの相互独立性を提唱している。本研究では，それぞれを実現する記述形式を設計した一方で，問題定義と数理モデルの対応づけを実現する記述形式を独立的かつ具体的に定義した。これによって，OR世界における一つの問題定義に対して異なる数理モデルを構築することと，一つの数理モデルを複数の問題定義に適用することが，必要に応じて柔軟にできる。

結合条件の記述形式では，相互に独立性を持つ結合子を用いて，対応する問題定義の要素と数理モデルの要素の結合を実現する。これによって，すでに対応づけられている問題定義と数理モデルに対しても，必要に応じて，異なる結合子の組合わせで異なる結合条件型を定義して，異なるレベルの問題定義と数理モデルの結合を実現できるであろう。

(4)弾力性に富む数理モデルの記述形式

モデル管理の研究が進めば，実行型のモデリング言語が要求される。一般的に，実行型のモデリング言語が数理モデルをもとにして研究され，その実現には数理モデルを記述する形式の数学的情報の記述能力，記載されている数学的情報の分解能力を前提条件としている。記述能力とは数理モデルに現れる数学的情報を完全に記載できることであり，分割能力は記述されている情報を必要に応じて分離し処理できることを指す。本研究のGERMを援用して設計した問題定義と数理モデルの記述形式では，問題定義の記述形式によって現実問題の物理的な意味を詳細に記述し参照できる一方で，そこから独立された数理モデルにおいて，数理モデルに含まれる数学的情報が構造化され分離可能な形式で記述される。数理モデルにおける数式の表記法そのものには言及していないが，それには記載されている数学的情報を利用して実行型のモデリング言語へを発展するか，あるいはAMPLなどの既存のモデリング言語を利用するかが，いずれも可能であると思われる。

(5)記述形式と事例ベースのインプレメンテーションにおけるデータ構造の関係

本稿ではGERMによる問題記述に含まれる様々なデータないし情報を整理して記述形式を定義し，それを用いて生産スケジューリング事例ベースの構築を検討した。しかし，コンピュータでの事例ベースのインプレメンテーションにおけるデータ構造は一般にこの記述形式と異なったものになる。

事例ベースのインプレメンテーションにおけるデータ構造はシステムの構築のために援用する実装のアプローチと手法，利用されるプログ

ラミング言語などに大きく依存する。例えば，オブジェクト指向アプローチを援用して事例ベースをインプレメントするなら，各種のオブジェクト（システム型，実体型，関連型，汎システム型など）を単位として必要なデータを抽象化して取り扱う。これに対して，関係データベースを利用して事例を記述すれば，記述形式における様々なデータないし情報はその種類によって，異なる関係表に格納するかも知れない。本稿で提案した記述形式は，問題定義／数理モデルの記述に関する基本的な情報枠組みである。これを利用して事例ベースなどのようなモデル管理システムをインプレメントするとき，必要に応じて適当なデータ構造に変換することが必要であろう。

6．結論

事例ベース方式によるモデル管理の前提となる問題定義，数理モデル構築の支援を検討し，構造化しモジュール化した問題定義／数理モデルの記述形式を提案し，それを生産スケジューリング事例ベースの構築に効果的に活用することを検討した。

モデリング・ライフサイクルの支援としては，モデルの解法開発，問題解決，モデル解釈などの支援も要求されるが，本稿では検討しなかった。また，事例ベース推論も検討しなかった。それらは今後の研究課題である。しかし，本稿で提案した問題定義／数理モデル記述の記述形式を用いれば，無理なくこれらの課題の研究へと展開できると思われる。

謝　辞

本研究の推進にあたって指導をしていただいた北海道大学経済学部関口恭毅教授に心から厚くお礼申し上げたい。また，同学部木村俊一教授には本稿執筆の最終段階で貴重なコメントをいただいた。記して感謝したい。

参考文献

[1] Banerjee, S. and Basu, A., (1993), "Model Type Selection in An Integrated DSS Environment," *Decision Support Systems*, Vol. 9, No. 1, pp. 75-89.

[2] Bharadwaj, A., Choobineh, J., Lo, A. and Shetty, B., (1992), "Model Management Systems: A Survey, "*Annals of Operations Research*, Vol. 38, No. 1-4, pp. 17-67.

[3] Blanning, R. W., (1986), "An Entity-Relationship Approach to Model Management," *Decision Support Systems*, Vol. 2, No. 1, pp. 65-72.

[4] Blanning, R. W., (1993),"Model Management Systems," *Decision Support Systems*, Vol. 9, No. 1, pp. 9-18.

[5] Chang, A. M., Holsapple, C. W. and Whinston, A. B., (1993),"Model Management Issues and *Directions*," *Decision Support Systems*, Vol. 9, No. 1, pp. 19-37.

[6] Chen, P. P., (1976),"The Entity-Relationship Model－Toward a Unified View of Data," *ACM Translations of Database Systems*, Vol. 1, No. 1, pp. 9-36.

[7] Choobineh, J., (1991),"A Diagramming Technique for Representation of Linear Models," *Omega, Int. J. Manag. Sci.*, Vol. 19, pp. 43-51.

[8] Codd, E. F., (1970),"A Relational Model for Large Shared Data Banks,"*Communications of the ACM*, Vol. 13, pp. 377-387.

[9] Conway, R. W., Maxwell, W. L. and Miller, L.W.,(1967),"*Theory of Scheduling* (Reading, Mass: Addison-Wesley).（関根　智明(1971)「スケジューリングの理論」，日刊工業新聞社)。

[10] Fourer, R., Gay, D. M. and Kernighan, B, W., (1990),"A Modeling Language for Mathematical Programming,"*Management Science*, Vol. 36, No. 5, pp. 519-554.

[11] Geoffrion, M., (1987),"An Introduction to

Structured Modeling,"*Management Science*, Vol. 33, No. 5, pp. 547-588.

[12] Geoffrion, M.,(1989),"The Formal Aspects of Structured Modeling,"*Operations Research*, Vol. 37, No. 1, pp. 30-51.

[13] Graham, R. L., Lawler, E. L.,Lenstra, J. K. and Rinnooy Kan, A. H. G., (1979),"Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey," *Annals of Discrete Mathematics*, Vol. 5, 287-326.

[14] Johnson, S. M., (1954),"Optimal Two- and Three-Stage Production Schedules With Setup Time Included,"*Naval Research Logistics Quarterly*, Vol. 1, pp. 61-68.

[15] Kolodner, J. L., (1993),"Case-Based Reasoning,"*Morgan Kuafmann Publishers Inc.*

[16] Lee, J. K. and Kim, M. Y.,(1995).Knowledge-Assisted Optimization Model Formulation:UNIK-OPT,"*Decision Support Systems*, Vol. 13, pp. 111-132.

[17] Liang, T. P., (1993),"Analogical Reasoning and Case-Based Leaning in Modeling Management Systems," *Decision Support Systems*, Vol. 10, pp. 137-160.

[18] Ma, J.,(1995),"An Object-Oriented Framework for Model Management,"*Decision Support Systems*, Vol. 13, pp. 133-139.

[19] MacCarthy,B.L. and Liu, J., (1993),"Methods in Production Scheduling,"*Int. J. Prod. Res.*, Vol. 31, No. 1, pp. 59-79.

[20] Müller-Merbach, H., (1983),"Model Design Based on the Systems Approach,"*J. Opnl. Res. Soc.*, Vol. 34, No. 8, pp. 739-751.

[21] Müller-Merbach, H., (1990), "Database-Oriented Design of Planning Models,"*IMA Journal of Mathematics Applied in Business & Industry*, Vol. 2, pp. 141-155.

[22] Nagar, A., Haddock, J. and Heragu., (1995a),"Multiple and Bicriteria Scheduling: A Literature Survey," *European Journal of Operational Research*, Vol. 81, No. 1, pp. 88-104.

[23] Nagar, A., Heragu, S. S. and Haddock, J., (1995b),"A Branch-and-Bound for Two-Machine Flowshop Scheduling Problem," *Journal of the Operational Research Society*, Vol. 46, No. 6, pp. 721-734.

[24] 関口 恭毅 (1992)「ＣＡＭＰ：順序づけ分枝限定アルゴリズム設計支援システム」, オペレーションズ・リサーチ, Vol. 37, No. 11, pp. 550-554.

[25] 関口 恭毅 (1994)「順序づけ問題の記述のための汎実体－関連アプローチの開発と検討」, 北海道大学経済学部ディスカション・ペーパー・シリーズB, No.12.

[26] 関口 恭毅 (1996)「問題定義の一方法：実体－関連アプローチの拡張」, 北海道大学経済学部ディスカション・ペーパー・シリーズB, No.18。

[27] 鮑 金源, 関口 恭毅 (1995)「スケジューリング問題のモデル・データの構造分析」, 精密工学会北海道支部学術講演会・講演論文集, pp. 91-92.

[28] 松本 和男 (1994)「生産スケジューリングへの熱い期待」計測と制御, Vol. 33, No. 7, pp.531-532.