新しい時間計算モデルとスケジューリングへの応用

田辺　誠

1　研究のねらい

我々は、普段の生活で以下のような「時間のやりくり」に関する問題に直面することがある。

例 1 (家を建てる) 大工 *1* 人が家を *1* 軒建てるのに *1*ヶ月かかる。

- 家を *2* 軒建てるには何ヵ月かかるか？
- *1*ヶ月で *2* 軒建てるためには何人の大工が必要か？

例 2 (朝食を作る) レシピによると、みそ汁と卵焼きを作るには以下の時間と道具が必要である。*2* 口コンロを持つ独身者（コック *1* 名）が、起床後 *10* 分で朝食にありつけるだろうか？

	みそ汁	卵焼き
下準備	*1* 分 *(*コック*)*	*1* 分 *(*コック*)*
調理	*5* 分 *(*コンロと鍋*)*	*4* 分 *(*コックとコンロとフライパン*)*
盛付け	*1* 分 *(*コック*)*	*1* 分 *(*コック*)*

これらの問題に対して我々はカレンダーや時計などの時間を感じる道具を身につけ、「締めきり」「時間切れ」「光陰矢の如し」などの時間を語る言葉を用いて対処している。

コンピュータの世界でも、原子炉の制御システムや列車の運転制御システムなどの、動作を時間内に完了することが求められる実時間システムでは、同じように時間を上手にやりくりすることが求められる。

従来、このような状況におけるプログラミングでは、プログラムの行ないやすさや保守のしやすさよりも計算の実行効率が重要視されて来た。課せられた仕事に対してコンピュータの計算能力に余裕がない場合、人よりもコンピュータにわかりやすい言葉を用いる必要があったのである。

しかしながら、安価で高性能なコンピュータが手に入る現在、計算環境に多少余裕のある場合は少なくない。この場合、プログラムの効率よりも保守性や再利用性に重点を置くことができるため、コンピュータよりも人に易しいプログラミング環境を整備することが課題となる。

このような環境を構築するためには、まず、時間に関する性質を表現するための、あいまいさのない言語体系をコンピュータに与える必要がある。さきがけ研究では、時間に関する我々の感覚を形式化することにより、これをコンピュータと共有することを目標とした。

2 研究成果

計算が成功するかどうかが、計算結果の値だけではなく計算を行なうタイミングにも依存する場合、この計算を実時間計算と呼ぶ。例えば「原子炉内の温度が1000度以上になったら0.2秒以内に制御棒を引き抜く」ための計算は実時間計算であり、「1から100まで順に足す」計算は実時間計算ではない。実時間計算の成功の可否は、プログラムの字面だけではなく、与えられた時間制約や利用可能な資源にも依存する。家を建てる場合、「大工2人」という資源を用意できるかどうかによって「1ヶ月以内に家を2軒建てる」という実時間計算が可能であるかどうかが決まる。実時間システムは一般に複数の実時間計算を同時に行なっているため、どの計算にどの資源をどれだけの時間割り当てるかを決めること、すなわちスケジューリングをうまく行なうことが計算を成功させるための大きな鍵を握っている。

時間に関する見方や切口にはいろいろ考えられるが、ここではスケジューリングに応用可能な研究成果として、さきがけ21で発展させた以下の2つの体系について説明する。

2.1 「時は金なり」 – 時間と資源を扱う論理系 –

時相線型論理 さきがけ研究では、計算に必要な時間と資源との関係を表現し、推論するための論理体系として時相線型論理を構築した。まず、計算に必要な時間と資源との関係を、論理式間の関係を表す式（シーケントと呼ばれる）によって表現した。例えば、「大工が1ヶ月で家を1軒建てることができる」という性質は、時相線型論理のシーケントによって以下のように表現できる。

$$\text{大工} \vdash [1\text{ヶ月}](\text{家} \otimes \text{大工})$$

また、シーケントから別のシーケントを導き出す推論規則(図1)を用いることにより、上記のシーケントと「家$\vdash [True]$家」(家はいつまでたっても壊れない)というシーケントから以下のシーケントを導くことができる。

$$\text{大工} \vdash [1\text{ヶ月} \otimes 1\text{ヶ月}](\text{家} \otimes \text{家} \otimes \text{大工})$$
（大工1人が家を2軒建てるには2ヶ月必要）

$$\text{大工} \otimes \text{大工} \vdash [1\text{ヶ月}](\text{家} \otimes \text{家} \otimes \text{大工} \otimes \text{大工})$$
（1ヶ月で家を2軒建てるには大工2人が必要）

$$\frac{\Gamma \vdash [X]A \quad \Delta \vdash [X]B}{\Gamma, \Delta \vdash [X](A \otimes B)}\ [\vdash \otimes] \qquad \frac{\Gamma, [X]A \vdash C}{\Gamma, [X](A \,\&\, B) \vdash C}\ [\mathrm{l}\& \vdash]$$

$$\frac{\Gamma \vdash [X]A \quad \Gamma \vdash [X]B}{\Gamma \vdash [X](A \,\&\, B)}\ [\vdash \&] \qquad \frac{\Gamma \vdash [Y]A \quad \vdash_R X \to Y}{\Gamma \vdash [X]A}\ [\mathrm{cut}_R]$$

$$\overline{[X][Y]A \equiv [X \otimes Y]A} \qquad \overline{[X \,\&\, Y]A \equiv [X]A \,\&\, [Y]A}$$

図 1: 時相線型論理の導出規則（抜粋）

時間ペトリネットのスケジューリング　時間ペトリネットは実時間並列計算の数学モデルであり、実時間計算の実行をシミュレートすることができる．時間ペトリネットによるシミュレーションと時相線型論理によるシーケントの導出との間には以下の関係があることがわかった。

時相線型論理	時間ペトリネット
論理式 A	シミュレーションの途中状態に関する性質
シーケント $A \vdash B$	「A を満たす状態で実行を終了できる時間ペトリネットは、スケジューリングを変更することによって B を満たす状態で実行を終了できる」という言明
$A \vdash B$ の導出	A を満たす状態で実行が終わるスケジューリングが具体的に与えられた時、これを B を満たす状態で実行が終わるスケジューリングに作り変える具体的な方法

両者にこのような密接な関係があることを利用して、実時間システムのスケジューリングに時相線型論理を応用できる。

ステップ 1 時間ペトリネットを用いて実時間システムのプロトタイプを作成する。

ステップ 2 作成されたペトリネットの部品（計算の 1 ステップに対応）に対応する時相線型論理のシーケントを求める。

ステップ 3 ステップ 2 で用意されたシーケントと時相線型論理の推論規則を用いて、望ましい性質に対応するシーケントを導出。

ステップ 4 ステップ 3 での導出の手順に対応して、望ましい性質を持つ時間ペトリネットのスケジューリングが定まる。

2.2 パラレル・ワールド – 複数の時間の流れを扱うプロセス代数 –

食事を時間内に準備する課程は、実時間計算の典型的な例を示している。朝食を作る例では「みそ汁を作る」「卵焼きを作る」という2つの実時間計算が要求され、どちらも「10分以内に完了する」という時間制約を持つ。

このような状況をモデル化するために作られた、ACSRと呼ばれるプロセス代数では、みそ汁だけを作る状況は

$$\text{みそ汁} = \{\text{コック}\}^{1\text{分}} : \{\text{コンロ}, \text{鍋}\}^{5\text{分}} : \{\text{コック}\}^{1\text{分}} . \overline{\text{みそ汁 } ok}$$

と記述できる。しかし、この式は「みそ汁を休みなく作り続け7分後に調理が完了するプロセス」を表しているため、卵焼きとの兼ね合いで調理が中断される可能性を考慮にいれる場合、

$$\begin{aligned} \text{みそ汁}_0 &= \overset{(\text{何もしないで1分待つ})}{\{\}^{1\text{分}} : \text{みそ汁}_0} \overset{\text{または}}{+} \overset{(\text{下準備をして次のステップへ})}{\{\text{コック}\}^{1\text{分}} : \text{みそ汁}_1} \\ \text{みそ汁}_1 &= \{\}^{1\text{分}} : \text{みそ汁}_1 + \{\text{コンロ、鍋}\}^{1\text{分}} : \text{みそ汁}_2 \\ &\cdots \end{aligned}$$

のように、中断の可能性をすべて記述する必要がある。

みそ汁のレシピ作成者の頭の中では、みそ汁を調理している間しか時間は経過しない。しかし，調理のプロセスを物理的な時間軸に沿って並べると，「みそ汁から見た時間」と「卵焼きから見た時間」はゲームがポーズボタンで中断されるのと同じように、時間軸上に寸断されて存在している（図2)。

ACSRによってレシピを記述する際の繁雑さは，モデル化の時点で「みそ汁の時間」「卵焼きの時間」をすべて物理的な時間軸に並べ直すことから来ている。

観察者によって時間の経過が違って見える状況を表すためにACSRの拡張を行なった(図3。ただし正確にはACSRと拡張ACSRは共に優先度に関する記述を含むが，本稿ではこれらを省略した)。この体系では、「[]」に囲まれたプログラムは独自の時間軸を持ち、「[]」の外側から「ポーズボタン」を押して内部の時間を止めることができる。朝食のレシピは以下のように表される。

$$\begin{aligned} \text{朝食レシピ} &= \text{タイムキーパー} \parallel [\text{みそ汁}] \parallel [\text{卵焼き}] \\ \text{タイムキーパー} &= \{\}^{10\text{分}} : \text{みそ汁 ok.卵焼き ok} \\ \text{みそ汁} &= \{\text{コック}\}^{1\text{分}} : \{\text{コンロ}, \text{鍋}\}^{5\text{分}} : \{\text{コック}\}^{1\text{分}} . \overline{\text{みそ汁 } ok} \\ \text{卵焼き} &= \\ &\quad \{\text{コック}\}^{1\text{分}} : \{\text{コック}, \text{コンロ}, \text{フライパン}\}^{4\text{分}} : \{\text{コック}\}^{1\text{分}} . \overline{\text{卵焼き } ok} \end{aligned}$$

この式を図3の規則にしたがって遷移させることで、調理のシミュレーションを行なうことができる。

拡張されたACSRのスケジューリングへの応用として、例えば

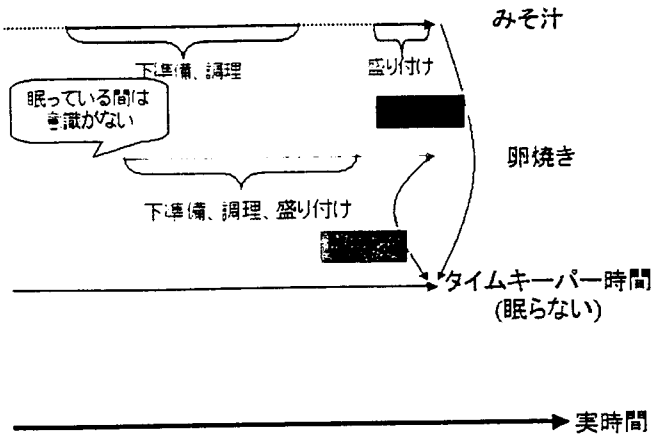

図 2: 朝食のスケジューリング

$$\frac{}{[P] \xrightarrow{\{\}} [P]} \qquad \frac{P \xrightarrow{\alpha} P'}{[P] \xrightarrow{\alpha} [P']} \qquad \frac{}{C{:}P \xrightarrow{C} P} \qquad \frac{}{a.P \xrightarrow{a} P}$$

$$\frac{P \xrightarrow{C} P' \quad Q \xrightarrow{C'} Q'}{P \parallel Q \xrightarrow{C \cup C'} P' \parallel Q'} \qquad \frac{P \xrightarrow{a} P' \quad Q \xrightarrow{\overline{a}} Q'}{P \parallel Q \xrightarrow{\tau} P' \parallel Q'} \qquad \frac{P \xrightarrow{\alpha} P'}{P \parallel Q \xrightarrow{\alpha} P' \parallel Q} \qquad \frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'}$$

C はリソースの集合。α はイベント a もしくはリソースの集合 C

図 3: 拡張 ACSR の導出規則（簡略化したものを抜粋）

- P:6 秒間に 1 回の実行を繰り返す。1 回当たりの計算時間は 3 秒
- Q: 3 秒間に 1 回の実行を繰り返す。1 回当たりの計算時間は 1 秒

の 2 つのタスクからなる実時間計算のスケジューリングをシミュレートすることができる（図 4）。

3 今後の展開

この研究を始めたいと思ったきっかけは、既存の計算モデルが扱う「時間」が自分の持つ時間の感覚と異なっていると感じたことであった。さきがけ研究の 3 年間では、「資源

$$
\begin{array}{lll}
 & D_p \parallel D_q \parallel [P] \parallel [Q] & \\
\xrightarrow{\tau} & (\{\}^6 : D_p) \parallel (\{\}^3 : D_q) \parallel [\{cpu\}^3 : P] \parallel [\{cpu\}^1 : Q] & \\
\xrightarrow{\{cpu\}^1} & (\{\}^5 : D_p) \parallel (\{\}^2 : D_q) \parallel [\{cpu\}^3 : P] \parallel [Q] & \text{(Q の計算を 1 秒間行なう)} \\
\xrightarrow{\{cpu\}^2} & (\{\}^3 : D_p) \parallel (D_q) \parallel [\{cpu\}^1 : P] \parallel [Q] & \text{(P の計算を 2 秒間行なう)} \\
\xrightarrow{\tau} & (\{\}^3 : D_p) \parallel (\{\}^3 : D_q) \parallel [\{cpu\}^1 : P] \parallel [\{cpu\}^1 : Q] & \\
\xrightarrow{\{cpu\}^1} & (\{\}^2 : D_p) \parallel (\{\}^2 : D_q) \parallel [\{cpu\}^1 : P] \parallel [Q] & \text{(Q の計算を 1 秒間行なう)} \\
\xrightarrow{\{cpu\}^1} & (\{\}^1 : D_p) \parallel (\{\}^1 : D_q) \parallel [P] \parallel [Q] & \text{(P の計算を 1 秒間行なう)} \\
\xrightarrow{\{\}^1} & (D_p) \parallel (D_q) \parallel [P] \parallel [Q] & \text{(アイドリング)}
\end{array}
$$

ただし

$$
\begin{array}{llll}
P & = & \mu X.\overline{s}.\{cpu\}^3 : X & Q = \mu X.\overline{s'}.\{cpu\} : X \\
D_p & = & \mu X.s.\{\}^6 : X & D_q = \mu X.s'.\{\}^3 : X
\end{array}
$$

図 4: 実時間タスクのシミュレーション

としてとらえることのできる時間」「複数の流れを持つ時間」という観点から基礎理論の研究を行なった。

研究成果は、時相線型論理、拡張 ACSR、時間イベントパターンなどの独立した理論的成果物からなる。理論のそれぞれが時間に対する別の見方を表すが、これらの相互の関連を体系付け、デザインパターンの時間に関するカタログ集を作ることが将来的な目的である。

そのためにまず、個々のカタログをしっかりしたものにする必要がある。3 年間で研究してきた理論をさらに深めながら、教科書の練習問題レベルを越えたプログラミングの現場への応用に地道に取り組んでいく。

謝辞 領域総括の安西先生を始めアドバイザーの先生方には、「役に立つのかな？」と心配していただきながらも暖かいご指導をいただきました。また、科学技術振興事業団の方々には、3 年間素晴らしい研究環境を与えていただきました。特に、領域事務所の方々には、時間に関する研究をしながらも提出物の締めきりをしばしば破るなどの数々の粗相にもかかわらず、暖かく支えていただきました。どうもありがとうございました。

4 成果リスト

- 泉田 大宗, 川勝 則孝, 田辺 誠, 中島 玲二, and 林 良生. 時間イベントパターン. **コンピュータソフトウェア**, Vol.17, No.5, 2000.

- 田辺 誠. リアクティブシステムの観察:直観主義様相論理によるアプローチ. *SLACS:* **記号論理と情報科学の研究集会**, 1998.
- 泉田 大宗, 川勝 則孝, 田辺 誠, 中島 玲二, and 林 良生. Recognizing timed event sequences – formalism, machines and applications – (extended abstract). **日本ソフトウェア科学会第** *15* **回大会**, 1998.
- M. Tanabe. Media object semantics for temporal linear logic. *MMM'97, the 4th International Conference of MultiMedia Modeling.* World Scientific, 1997.

計算モデルの新展開

米 崎 直 樹

従来の計算モデルは、値を結果として出力する計算を主たる対象としていたが、システムの制御や、そこで用いられる通信プロトコル、オペレーティングシステムなど、今日の計算機上で動く多くのソフトウェアは、その時間的な振る舞いにその計算の意味の本質を持っている。

これらの応用領域の多くは、交通やエネルギー供給システムにとどまらず、最近では電子商取引システムの分野などで、すでに社会的インフラの重要部分を占めるようになっており、設計ミスによる異常動作が無いことや、悪意を持った第3者からの攻撃に対して耐性があることなどが、社会の安全と安定のために必須の要件となって来ている。

これらの問題に対処するためには、形式手法と呼ばれる厳密な計算モデルに基づいたシステムの構築方法が重要となっており、その形式的取り扱いをめぐって、様々な計算モデルが提案されている。

ここでは、最近のその取り扱い方法を概観、比較すると共に、それらを用いてシステムを記述した際の計算機による自動的設計支援等について、様々な観点から現状を総括すると共にその将来的発展を展望する。この中で田辺氏の提案する時間線形論理による取り扱いや、複数の時間の流れを扱うプロセス代数 ACSR、そして時間イベントパターンの考え方の位置付けを行う。

すなわち、論理による方法、代数的手法そしてオペレーショナルなモデルを用いる方法間の関係とトレードオフについて述べる。特にその実行、検証、プログラム自動生成における、無限状態空間からの抽象化による有限状態空間への帰着、有限であるが状態数爆発することへの対処法、パラメター化による一般化の手法やその現実的問題への適用状況について述べ、将来的な研究方向を提示する。