Vol.2020-MPS-127 No.2
2020/3/2

多重 Ambient Calculus のための統合開発環境

加藤 暢[a)]　山田 瞭太　樋口 昌宏

概要：本発表では，多重 Ambient Calculus (MAC) のための統合開発環境を提案する． MAC とは，物流システムのモデル化に特化したプロセス代数である．コンテナを用いる海上物流では，多数のコンテナが数隻のコンテナ船に載せ替えられながら，何箇所かのハブ港を経由し目的地に輸送される．我々は現在，コンテナ輸送が計画通りに行われているかどうかを確認する物流監視システムについて研究を進めている．このシステムは，MAC を用いてモデル化された物流計画と，そのモデルの一部を書き込んだ RFID タグを用いて検知した実際のコンテナの移動を比較することで，コンテナが物流計画通りに取扱われているかどうかを確認するものである．このような監視活動を行うためには，物流システムの持つ動的な階層構造を表現する MAC のプロセス式を正確に記述する必要がある． 本発表で提案する統合開発環境 IDE4MAC は，通常の編集機能に加え，複数の動作が選択可能なプロセスに対する選択実行機能や巻き戻し機能など，MAC のプロセス式の持つ全ての非決定的な動作を確認するために有用な機能を備える．さらに，プロセス式の階層構造を表す木構造の図を編集する機能，及び木構造の図とプロセス式のリアルタイムな相互変換機能を備える．IDE4MAC を使用することで，大規模かつ複雑な経路をもつ輸送システムのモデル化が容易になり，物流監視システム開発の効率を上げることが可能となる．IDE4MAC は，インストールや使用の利便性を考慮し，エクリプスのプラグインとして開発している．

キーワード：統合開発環境，プロセス代数，Ambient Calculus，海上物流システム

The Integrated Development Environment for the Multiple Ambient Calculus

TORU KATO[a)]　RYOUTA YAMADA　MASAHIRO HIGUCHI

Abstract: We propose an integrated development environment (IDE) for developing formulae written in the multiple ambient calculus (MAC). MAC is a kind of process algebra designed for modeling freight systems in which a lot of containers are transported by several vessels from one port to another port via several hub ports. We have been studying a handling management system that confirms the validity of container handling during shipping. The system checks whether containers are being correctly handled by comparing the movement of the containers, which is sensed by radio frequency identification (RFID) tags, with formal models (formulae) written in MAC. For accurate and automatic management, we need to develop MAC formulae that accurately express the nested and dynamic changing structure of the entire freight system (ports, vessels, and containers) and the movement of those objects. The IDE presented in this paper (IDE4MAC) equips ordinal editing functions and several debugging functions such as a selective execution function and backward tracing function that are useful for checking all non–deterministic actions of MAC processes. IDE4MAC also has a graphical editor by which we can edit tree structures expressing nested structure of MAC processes and the real-time mutual conversion function between process formulae and tree structures. IDE4MAC enables us to easily model freight systems with large scale and complicated paths, that improves the efficiency of developing the handling management system. We developed IDE4MAC as plugins of Eclipse so that it can be easily installed and used.

Keywords: IDE, process algebra, Ambient Calculus, freight system

1. はじめに

本発表で提案する統合開発環境 (IDE4MAC) は，現在研究を進めている物流監視システム [11] で使用する多重 Ambient Calculus(MAC) [4] のプロセス式設計支援を主な目的として開発されたものである．この物流監視システムは，物流システム全体を MAC のプロセス式でモデル化し，RFID タグに書き込んだプロセス式の遷移と，RFID 機器で読み取った実際のコンテナの動きを比較することで，コンテナが輸送計画通りに扱われているかどうかを自動的に確認するものである．

MAC が基礎としている Ambient Calculus(AC) [1] は，動的な階層構造を代数式で表現することができ，ネットワーク間を移動するようなモバイルプロセスの記述に特化したプロセス代数である．一方海上物流にも，大きな枠組みであるコンテナ船が小さな枠組みであるコンテナを積んでいるという階層構造が存在している．そしてその階層構造は，船がある港から別の港へと移動したり，コンテナの積込みや積降ろしをするとき動的に変化する．

我々は AC と物流システムの双方が持つ動的な階層構造という類似点に着目し，文献 [7] において，物流書類の内容を AC のプロセス式を用いて表現し，実際の物流がそのプロセス式の記述どおりに行われているかを監視するシステムを提案した．さらに，現実の物流で頻繁に生じるコンテナの追加，キャンセルなど動的な変更にも柔軟に対応できるモデル化を行うため，我々は文献 [4] において MAC を提案し，文献 [2] や文献 [11] において MAC を用いた物流監視システムを提案した．これらのシステムは，エクセル形式で記述された物流書類から MAC のプロセス式を生成する機能，MAC の式を遷移させることのできる処理系，及び RFID 機器を用いて検知した実際のコンテナの取扱いをプロセス式の遷移と対比させ，その取扱いが正しいものであるかどうかを監視する機能からなる．

このような監視活動を行うためには，物流システムの持つ動的な階層構造を表現する MAC のプロセス式を正確に記述する必要がある．IDE4MAC は，通常の編集機能に加え，複数の動作が選択可能なプロセスに対する選択実行機能や巻き戻し機能など，MAC のプロセス式の持つ全ての非決定的な動作を確認するために有用な機能を備える．さらに，プロセス式の階層構造を表した木構造の図を編集する機能，及び木構造の図とプロセス式のリアルタイムな相互変換機能を備える．IDE4MAC を使用することで，大規模かつ複雑な経路をもつ輸送システムのモデル化が容易になり，物流監視システム開発の効率を上げることが可能となる．

[1] 近畿大学理工学部情報学科
Kindai University, Kowakae, HigashiOsaka 3–4–1, Japan
[a)] kato@info.kindai.ac.jp

本発表ではまず 2 章で MAC による物流システムのモデル化の例を示す．3 章では 本研究で提案する IIDE4MAC の持つ各種機能について説明する．4 章では IDE4MAC を用いたプロトコル検証の実例について述べ，5 章では IDE4MAC がある程度の規模のプロセス式生成にも耐えられることを説明する．7 章ではプロセス代数の統合開発環境として有名な PAT と IDE4MAC を比較する．

2. 多重 Ambient Calculus(MAC)

本研究では文献 [4] で定義された MAC の構文規則と遷移規則そのまま利用する．本章では，物流システムの簡単なモデルを用いて，これらの規則を直観的に説明する．

2.1 MAC による物流システムのモデル例

AC による物流システムのモデル化では，コンテナ数が増えると同期制御のための記述が複雑になり，コンテナの追加やキャンセルといった現実の物流システムで頻繁に起こる変化に柔軟に対応することが困難となる．MAC ではこの問題を，複数のプロセス式の組で物流システムをモデル化することで回避している [3,4]．MAC の構文規則である全体式と個別式は文献 [1] で以下のように定義されている．

定義 2.1 (全体式と個別式 (MAC の構文規則))
$P_1, P_2, \ldots, P_n$ をそれぞれ *AC* のプロセスとする．この時 $\overline{P} = (P_1, P_2, \ldots, P_n)$ を全体式，各 $P_i (1 \leq i \leq n)$ を $\overline{P}$ の個別式，あるいは第 i 成分と呼ぶ． □

簡略化した例を用いて定義 2.1 を説明する．式 (1) は，2 つのコンテナ $co1$ と $co2$ を東京港 TK で船 $SHIP$ に積み込むという物流システムを表した全体式である．

$$
\begin{array}{l}
(\\
\quad TK[SHIP[co1[\ldots] \mid out\ TK.\ in\ HKG]] \mid HKG[\,], \\
\quad TK[co2[\ in\ SHIP.\ lcomp[out\ co2]] \\
\qquad \mid SHIP[open\ lcomp.\ out\ TK.\ in\ HKG]] \mid HKG[\,], \\
\quad TK[SHIP[\ out\ TK.\ in\ HKG]] \mid HKG[\,] \\
)
\end{array}
\tag{1}
$$

式 (1) の 2 行目にある第 1 成分は，すでに $SHIP$ に積み込まれているコンテナ $co1$ の動作を表わす個別式であり，3，4 行目にある第 2 成分は，積込み前のコンテナ $co2$ の動作をを表す個別式である．また 5 行目にある第 3 成分が東京港から香港港へ向かう船の航路を表す個別式である．このように MAC では，各コンテナ及び船ごとに個別式という独立した AC のプロセス式でそれぞれの性質を記述する．これにより，コンテナの追加やキャンセルを個別式の挿入や削除で表現することが可能となる．

MAC では，唯一つの個別式にのみ現れる ambient 及び制御用 ambient を**個別 ambient** と呼び，複数の個別式に共通して現れる，制御用 ambient 以外の ambient を**大域 ambient** と呼ぶ．式 (1) において，$co1$，$co2$ ambient はそれぞれ第

1 及び 2 成分である個別式にのみ現れる個別 ambient である．また *TK*，*HKG*, *SHIP* は複数の個別式に共通に現れる大域 ambient である．以降，大域 ambient 名は先頭に大文字を用いて区別する．

MAC の遷移規則については，文献 [4] の定義をそのまま利用する．ここでは式 (1) に示した全体式を用いて遷移規則を直観的に説明する．式 (1) の第 1 成分では *SHIP* の *out TK* という遷移が可能な状態であるが， 第 2 成分では *SHIP* の *out TK* の前にコンテナ積込み確認用の capability があるため *out TK* はブロックされている．第 2 成分で *co*2 ambient が *SHIP* に入り *SHIP* の *out TK* が可能になった時，全ての成分で *out TK* が可能になり，この時点で始めて *SHIP* の *out TK* という遷移が可能になる．このように MAC では，遷移規則によって全てのコンテナが積まれた後に船が出航すると言った同期的な動作を表現するため，AC のような複雑な記述は不要になる．

3. MAC のための統合開発環境

2.1 節で示したような MAC のプロセス式は，物流システムの構成やその中のモノの動きを直接的に表したものになっている．しかし，多くのコンテナが複数のハブ港を経由して複数のコンテナ船に載せ替えられながら輸送される様子をモデル化する式を emacs 等の一般的なエディタで正確に記述することは困難である．そのような作業を支援する目的で我々は IDE4MAC を構築した．IDE4MAC は，インストールや使用の利便性を考慮し，エクリプスのプラグインとして Java 言語で開発しているため，Windows，Mac OS そして Linux 上で動作する[*1]．

3.1 グラフィカルエディタ

もちろん IDE4MAC はプロセス式作成を支援するテキストエディタを持つが，IDE4MAC の最大の特徴は，図 1 に示すグラフィカルエディタ (GE) である．CCS や CSP，π 計算などの他のプロセス代数と異なり，AC および MAC は，プロセス式の [括弧] による階層構造がそのまま現実の対象物の構造を内部構造に至るまで表現している．従って，いかに正確にその階層構造を把握できるかが，MAC プロセスを記述するために重要となる．

テキスト形式のままでも階層構造の理解は不可能ではないが，それを木構造で表現することができれば，直観的に対象物の構造を把握し編集することが可能となる．また AC や MAC ではプロセス式の遷移にともなう階層構造の変化により対象物の動作を表現しているが，木構造の変化する様子を目視することにより，対象物の動作を直観的に把握することが可能となる．

IDE4MAC の GE は，このような木構造に対する直観的な操作を実現するため実装された．本節では GE の備える各種支援機能について説明する．

図 1–① の部分をキャンバスと呼ぶ．

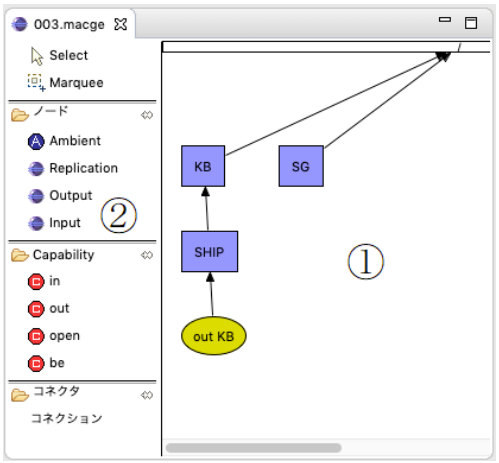

図 1 木構造編集用のグラフィカルエディタ
Fig. 1 Graphical editor for composing tree structures

この部分にプロセス式の階層構造を表現する木構造を記述する．四角いノードが ambient を，丸いノードが capability を表す．図 1–② の部分をパレットと呼ぶ．ここから ambient や capability をパーツとして取り出しキャンバスに置くことで，木構造を記述することができる．一般的に編集中は図 3 のようにノードが多数現れるが，特定の部分の編集に直接関係しない部分木は理解に邪魔なので，ノードをクリックすることでその下の部分木を収納・展開することができる．

3.2 TE–GE 連携機能

IDE4MAC では，図 2 に示すように TE と GE を同時に見比べながらプロセス式を記述することができる．これを我々は TE–GE 連携機能と呼んでいる．プロセス式を記述する際，片方のエディタ上でファイルが上書き保存された際に，もう一方のエディタへ変更が自動的に反映される．

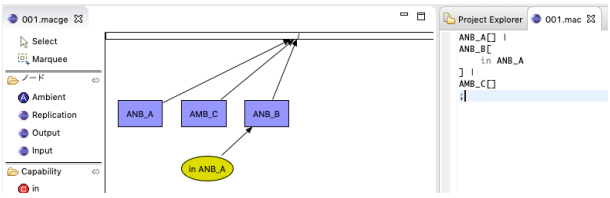

図 2 GE の上書き保存後
Fig. 2 Orverwirte saving on GE

3.3 GE 上のブレークポイント機能

GE 上の任意の capability を右クリックして現れるメニューから「Breakpoint」を選ぶことで，選択した capability を消費するまで MAC の遷移規則に基づいた木構造上での

[*1] ただし最新の Eclipse では動作せず，現状 Eclipse IDE for Java EE Developers Luna のみで動作を確認している．

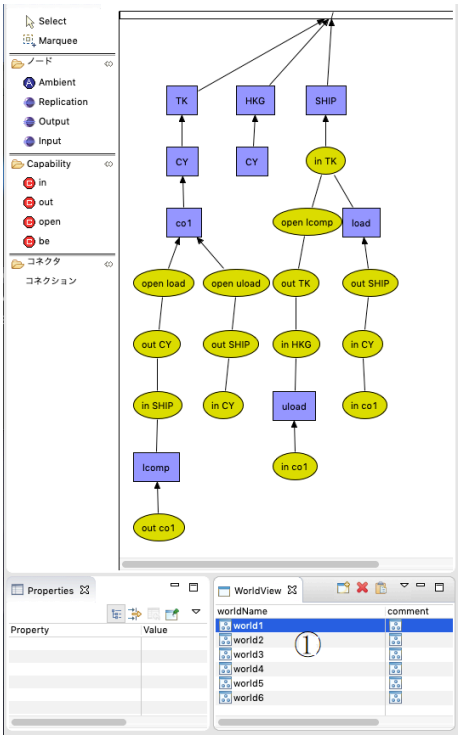

図 3 *co*1 の個別式
Fig. 3 Indivisual expression of *co*1

遷移が行われる．また，任意の状態にまでワンステップずつ連続で戻す事が出来る巻き戻し機能も持つ．

3.4 GE 上の個別式切り替え機能

MAC が AC や他のプロセス代数と大きく異なるのは，2.1 節の式 (1) で示したように，個々の対象物の性質を個別式で表現し，それらの組合せで全体の動作を表現している点である．個別式それぞれの同期的な動作を組み合わて全体の同期的な性質，例えば全てのコンテナが積み込まれた後に船の出港が許可されるといった性質が表現される．従って，個々の個別式それぞれの同期的な性質を見比べながらプロセス式を作成する必要がある．

TE では個別式をコンマで区切って一括して表示するが，GE ではこの区切りを解析し，個別式をそれぞれ独立した木構造として表示する．図 3 は式 (1) の 2 行目に示したコンテナ *co*1 の個別式 (の詳細版) を表す木構造である．この図の① の部分で，全体式中に存在する全ての個別式を切り替えることができる．GE の持つこの機能により，各個別式を自由に切り替えそれぞれの構造を把握しながら，対象となるシステム全体をモデル化することが可能となる．

3.5 MAC の処理系

IDE4MAC では，我々が物流監視システムのために開発した MAC の処理系を利用してプロセス式が実行される．エディタで記載したプロセス式はこの処理系に渡され Java オブジェクトに変換され (図 4–①)，MAC のプロセス式の木構造を表現したオブジェクトの木構造として Java 仮想機械 (JVM) 上で管理される (図 4–②).

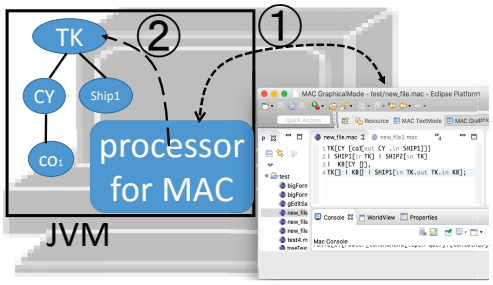

図 4 MAC 処理系
Fig. 4 Processor for MAC

4. 実用例

1 章で述べたように，我々が研究している物流監視システムでは，MAC のプロセス式で物流システムの動作を記述し，そのプロセス式を実際のコンテナの取扱の監視に利用している．IDE4MAC には PAT [8] のようなモデル検査機能は無いが，これまでに述べてきた機能を用いて，開発途中のプロセス式に対するプロトコル検証を行った実例について述べる．実際の物流システムでは様々な作業が同期的に行われる．その一例を以下に示す．

条件 1: 積下ろしにおける船の出港制限
船は積下ろし港において，予定されたコンテナの積下ろしが完了するまで出港してはならない．

条件 2: 積込みにおける船の出港制限
船は予定されたコンテナの積込みが完了するまで出港してはならない．

これらに反する動作が起こった場合警告を出すことが物流監視システムの役割の一つである．条件 1，2 はそれぞれ式 (2)，(3) で表すことができる (と考えていた)．これらの式に現れる *SG* はシンガポール港を表している．

$$\begin{aligned} &SG[\\ &\ SHIP\_1[open\ ulcomp.out\ SG.(\text{中略})] \\ &\ |CY[co1[ulcomp[out\ co1.out\ CY.in\ SHIP\_1]]] \\ &] \end{aligned} \tag{2}$$

式 (2) は，*SHIP_1* で運ばれてきたコンテナ *co1* が *SG* 港で積み降ろされ，*SG* 港のコンテナヤード *CY* に搬入された状態を表している． この後，*co1* 中の制御用 ambient *ulcomp* が *co1* 及び *CY* を出て *SHIP_1* に入り，*SHIP_1* 中の *open ulcomp* が消費されることによって *SHIP_1* 中の *out SG* が活性になり，*SHIP_1* が *SG* 港を出ても良いことを表現している．つまり制御用 ambient *ulcomp* が *SHIP_1* に対して，*co1* の積下ろしが完了したことを通知しているのである．

$$\begin{aligned} &SG[\\ &\ SHIP\_2[open\ lcomp.out\ SG.(\text{中略})] \\ &\ |CY[co1[out\ CY.in\ SHIP\_2.lcomp[out\ co1]]] \\ &(\text{以下略}) \\ &] \end{aligned} \tag{3}$$

式 (3) は，*SG* 港のコンテナヤード *CY* にあるコンテナ *co1* を今から *SHIP_2* に積み込もうとしている状態を表している．まず *co1* が *CY* を出て *SHIP_2* に入る．その後 *co1* の制御用 ambient *lcomp* が *co1* から出ることによって *SHIP_2* 中の *open lcomp* が消費され，*SHIP_2* 中の *out SG* が活性になり，*SHIP_2* が *SG* 港を出ても良いことを表現している．つまり制御用 ambient *lcomp* が *SHIP_2* に対して，*co1* の積込みが完了したことを通知しているのである．

式 (2)，(3) はそれぞれ単独では問題なく動作し，条件 1, 2 を適切に表現できているように思われていた．しかし IDE4MAC 上でこれら 2 つの式を合わせて *co1* の個別式として作成し動作させた場合，この通りに動く場合と，*SHIP_1* が出港せず，*SHIP_1* に載せられている他のコンテナが目的地に到着しない場合があることがわかった．原因は *co1* 内の *ulcomp* が *co1* を出る前に *co1* が *SHIP_2* に移動してしまい，*ulcomp* が *SHIP_1* に移動できなかったためである．これは AC の非決定的性質に起因する同期の失敗例であり，文献 [6] において grave interference と呼ばれている AC 特有の落とし穴であった．文献 [11] で構築したモデル検査システムでももちろん反例としてこの現象は捉えることができるが，正確な場所を特定し，対応策を提案することは困難であった．

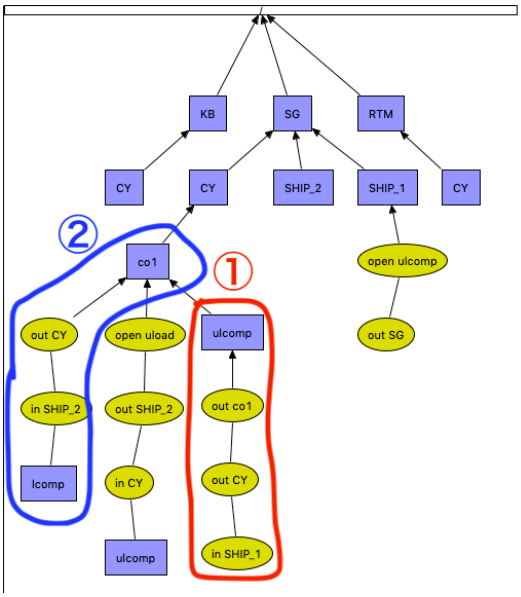

図 5　grave interference 発生場所の特定

図 6　The place where a grave interference occured

これに対し IDE4MAC の TE–GE 連携機能を用い，図 6 に示すようにこの状態の式と木構造を見比べながらブレークポイント機能と巻き戻し機能を使うことで，正しく動作する場合と grave interference が起きる場合の式の構造や実行順序を確認することが可能であった．この図の① が条件 1 を表した式 (2)，② が条件 2 を表した式 (3) の該当部分である．さらに，ブレークポイント機能と巻き戻し機能を使うことで *ulcomp* の移動を *co1* の移動よりも優先させるような式の作成を容易に行うことができ，*SHIP1* に載せられている他のコンテナが目的地に到着しないような実行系列を排除することが可能となった．

5. 大規模なプロセス式での動作確認

IDE4AC が大規模なモデル作成にも耐えられることを示すため，コンテナを 100 個，船を 3 隻，港を 4 つ使用する物流計画を表すプロセス式 (ノード数 6151) の作成及び実行を行った．このプロセス式は，神戸発シンガポール行の *SHIP_1*，シンガポール発ロッテルダム行の *SHIP_2*，ロッテルダム発ハンブルク行の *SHIP_3* を用いてコンテナ 30 個を神戸からロッテルダムへ，コンテナ 20 個を神戸からハンブルクへ，コンテナ 50 個をシンガポールからハンブルクへ輸送するという運送計画である．

下記の実験環境において，式や木構造の記述，相互変換などはタイムラグなく操作可能可能であり，全てのコンテナが仕出港から仕向港に達し遷移が完了するまでに掛かった時間は 4 分 35 秒であった．

モデル:　MacBook Pro (13-inch, 2017 年モデル)
プロセッサ:　2.3 GHz Intel Core i5
メモリ:　8 GB 2133 MHz LPDDR3
OS:　MacOS Mojave 10.14.6（18G2022）

6. 関連研究

プロセス代数に対する統合開発環境としては CSP [5] に対する PAT [8, 10], が有名である． PAT は単なる統合開発環境ではなく，様相論理を用いて記述された性質をプロセス式が満たすかどうかを確認するためのモデル検査機能を有する統合検証環境であり，PAT を用いたモデル検査に関する成果が多数報告されている．例えば文献 [9, 10] では，ネットワーク上に分散されたノードから代表元を選出する，いわゆるリーダー選出プロトコルを PAT で実装し，実装されたプロセスが常にただ一つのリーダーを選出できるかどうかを PAT で検証できることを示している．また文献 [17] では，UML のシーケンス図で表されたシステムを CSP に変換して PAT を用いてそのシステムに対するモデル検査を行うためのツールが提案されている．PAT に付属するチュートリアル [8] 内では，文献 [12] で提示されている列車運行管理システムの安全性を検証する例を PAT で記述し，PAT でもその安全性が検証できることを示している．これらの研究では，CSP で記述されたモデルに対し PAT でモデル検査を実施し，必要であれば反例を示して，モデル内部で所期の性質を満たさない部分を特定できることを示している． しかしモデルそのものを生成したり修正したりすることをいかに PAT が支援しているかについては述べられていない．PAT にはプロセス式と状態遷移図を見比べる機能はあるが，プロセス式の構造をグラフィカルに表示し

作成を支援する機能は持たない.

これに対し IDE4MAC は構造理解が重要な MAC のプロセス式とその木構造を見比べ，どちらを使っても式の遷移や修正が可能であるため，4 章で述べたように複雑な同期制御をもつ MAC のプロセス式の作成に有効であると思われる.

また 3.4 節で述べたように，MAC のプロセス式は他のプロセス代数と違い複数のコンポーネント (個別式) から構成されるため，各コンポーネントの木構造の表示を切り替えながらプロセス式の同期制御を設計できることも IDE4MAC の特徴となっている.

7. 結論

本発表では，多重 Ambient Calculus(MAC) のための統合開発環境である IDE4MAC を提案し，IDE4MAC のもつ物流システムのモデル化に有用な各種機能について説明した．物流システムには，動的に階層構造が変化し (船へのコンテナの積込みや船の入出港など)，さらに構造の構成要素が動的に増減する (コンテナの追加・キャンセルなど) という特徴がある．MAC はこの特徴をモデル化するために設計された言語である． 他のプロセス代数と違うこの特徴により通常のエディタでは難しかったプロセス式の作成に対し IDE4MAC の各種機能が有効に機能することを示した．とくに，AC に固有の同期的な動作の失敗原因である grave interference によって生じていたプロトコル設計の問題箇所と詳細の把握および修正が，IDE4MAC によって可能だったことを示した.

IDE4MAC はもちろん AC のプロセス式作成にも有効である．我々は現在，強化学習プログラム fineOptimAI を用いた組合せ最適化の研究を行っている [16]．この中で，強化学習の対象の一つである効率的なコンテナ荷役の問題に対し，我々は IDE4MAC を用いて作成したモデル (AC のプロセス式) を活用している．荷役問題を AC でモデル化し，AC の持つ非決定的な性質を用いて荷役の実行系列を書き出し，その実行系列を中心に学習をすすめることで，fineOptimAI の性能を向上させることが可能である．このように IDE4MAC は物流監視システムだけでなく，階層構造が動的に変化する様々な対象に関する問題に応用することが可能である.

謝辞 本研究は，JST START(ST191004QB) の支援を受けたものである.

参考文献

[1] Cardelli, L. and Gordon, A. D.: Mobile Ambients, *Theoretical Computer Science*, Vol. 240, pp. 177–213 (2000).
[2] 橋本隆弘，加藤暢，樋口昌宏：多重 Ambient Calculus と UHF 帯 RFID 機器を用いた海上物流監視システム，情報処理学会論文誌: プログラミング， Vol. 6, No. 2, pp. 1–12 (2013).
[3] 樋口昌宏，森田哲平，加藤暢：多重 Ambient Calculus による物流記述に対する弱双模倣等価性を用いたモデル検査，情報処理学会論文誌:プログラミング，Vol. 5, No. 3, pp. 56–60 (2012).
[4] 樋口昌宏，加藤暢：物流システム記述のための多重 Ambient Calculus，情報処理学会論文誌:プログラミング，Vol. 5, No. 2, pp. 79–87 (2012).
[5] Hoare, C.: *Communicating Sequential Processes*, Prentice Hall (1985).
[6] Levi, F. and Sangiorgi, D.: Controlling Interference in Ambients, *Proceedings of the 27th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, ACM, pp. 352–364 (2000).
[7] 森本大輔，加藤暢，樋口昌宏：Ambient Calculus を用いた物流検査システム，情報処理学会論文誌，Vol. 48, No. SIG 10(PRO33), pp. 151–164 (2007).
[8] project, P.: PAT: process Analysis Toolkit, https://pat.comp.nus.edu.sg.
[9] Si, Y., Sun, J., Liu, Y., Dong, J. S., Pang, J., Zhang, S. J. and Yang, X.: Model checking with fairness assumptions using PAT, *Frontiers of Computer Science*, Vol. 8, pp. 1–16 (2013).
[10] Sun, J., Liu, Y., Dong, J. S. and Pang, J.: PAT: Towards Flexible Verification under Fairness, *Proc. 21st International Conference on Computer Aided Verification (CAV'09)*, LNCS, Vol. 5643, Springer, pp. 709–714 (2009).
[11] 加藤暢，高岡久裕，樋口昌宏，大山博史：多重 Ambient Calculus を用いた動的な海上物流計画に対するモデル検査，情報処理学会論文誌: 数理モデル化と応用， Vol. 11, No. 3, pp. 84–99 (2018).
[12] Yi, W., Pettersson, P. and Daniels, M.: Automatic Verification of Real–Time Communicating Systems by Constraint Solving, *Proc of the 7th International Conference on Formal Description Techniques*, North-Holland, pp. 223–238 (1994).
[13] 国土交通省：メコン地域陸路実用化実証走行試験，http://www.mlit.go.jp/kisha/kisha07/15/151018/01.pdf (2007).
[14] 国土交通省：国土技術政策総合研究所資料 (海上輸送を中心とした最近のサプライチェインセキュリティの動向 (その 2))，www.nilim.go.jp/lab/bcg/siryou/tnn/tnn0585pdf/ks0585.pdf (2010).
[15] 国土交通省：コンテナ物流情報サービスシステム「Colins」の中国との連携について，www.mlit.go.jp/report/press/port02_hh_000053.html (2011).
[16] 平嶋洋一，加藤暢，青山正人，黒田規義：組合せ爆発を計算可能な小さな AI fineOptimAI（ファインオプティマイ）の事業化，科学技術振興機構大学発産業創出プログラム START, https://www.jst.go.jp/start/project/index.html (2019).
[17] 海津智宏，磯部祥尚，鈴木正人：SDVerifier:プロセス代数 CSP を用いたシーケンス図検証ツール，コンピュータソフトウェア， Vol. 32, No. 1, pp. 1_234–1_252 (2015).
[18] 東芝ロジスティクス：東芝ロジが異業種向けに次世代グローバル混載サービス拡大，カーゴニュース，No. 10 月 13 日, p. 1 (2015).
[19] 経済産業省：物流業界における電子タグ等の活用実証実験国際コンテナ輸送，www.mlit.go.jp/kisha/kisha05/10/100329_2/02.pdf (2005).