修 士 論 文

Ambient Calculus を用いた移動エージェントの形式化

指導教官　渡部卓雄 助教授

北陸先端科学技術大学院大学
情報科学研究科情報システム学専攻

峯下聡志

1999 年 2 月 15 日

要旨

データを伴いながらネットワーク上を移動し実行されるプログラムを移動エージェントという。この移動エージェントを設計する言語において、その働きを真に理解するために必要である意味論は重要であり、今後の研究課題となっている。本研究では、移動エージェントの形式化、特に通信の切断における計算モデルを提案し、Ambient Calculus を用いた操作的意味論でその意味の定義を行なった。これにより、通信の切断があった場合の移動エージェントの動作の意味を直観的な表現で表すことができ、いくつかの仕様を与えることができた。

目 次

第 1 章

序論

分散型計算機環境における研究が色々となされる中、分散型ソフトウェアの開発に用いられる分散プログラミング言語に関する研究、特に処理系の設計に大きく関わってくる部分である型や意味論の研究が重要となっている。そして、分散プログラミング言語の意味の厳密な記述を行うことにより、プログラムの効果と生み出される結果をより理解でき、分散ソフトウェアの開発にも大いに役に立つ。本研究では移動エージェント言語に注目し、切断時操作、フォールトトレランスなどの形式的モデル・仕様を与えることを目標とする。以下に、本研究の背景、目的、概要、及び本稿の構成を簡単に述べる。

1.1 背景

分散型計算機環境の出現により、この環境における処理を行なうソフトウェアが必要となってきた。そして、C／Sやエージェント等の分散型ソフトウェアの研究に伴い、これらを研究・開発したり、分散データベースシステムを取り扱うなど、分散環境のための言語を用いることとなる。それが分散プログラミング言語である。しかし、分散プログラミング言語は理論や設計など、まだ十分な研究がなされておらず、今後、大きな課題である分散処理において、分散プログラミング言語の型推論、意味論、処理系、設計などの研究が大変重要となってくる。

その中で形式意味論は、プログラミング言語の本質とその言語で書かれたプログラムの振る舞いを真に理解するために必要であり、またプログラムの性質の論証において重要であるため、処理系や設計に大きく関わってくるものである。このため、分散プログラミング言語における意味論の厳密な記述を行なうことが急務であり、この厳密な記述によって正当性の検証や言語設計の改良に役立ち、分散ソフトウェアの発展へとつながる。

また、本研究で行なう移動エージェントについては、切断時操作、フォールトトレランスといった様々な計算モデルが提案されてきているが、これを実現する分散プログラミング言語の意味論がまだ定義されておらず、今後の研究課題となっている。

1.2 目的

本研究の目的は、現実的な移動エージェントの諸動作の形式的モデル・仕様を与えることである。本研究では特に通信の物理的切断が起こった場合の移動エージェントの諸動作について考え、切断時操作、耐故障動作の２つの形式化を行うこととする。

切断時操作は、常に結果（途中結果）を返しながら作業をつづけるプロセスについて考える。また、フォールトトレランスは、物理的な切断が一時的なものか永続的なものかの判断が難しいため、カウントダウンを行ない、それが終了した時点において通信がとれない場合に非常用プロセスを作動させるものについて考える。

1.3 概要

本研究では、前節の目的に沿って、移動エージェントの切断時操作、物理的な切断が起こった場合のフォールトトレランスを定義する。

形式的意味を定義する方法として操作的意味論を用いる。また、calculus はプロセス・デバイスが ambient を持ったまま移動を行なうという概念をもとに考えられた Ambient Calculus[2] を用いる。Ambient Calculus は、π-calculus[4] の概念と手法を基礎としており、移動エージェントを表現する能力を十分に持つものである。

本研究では、主に通信の切断があった場合の移動エージェントの動作の意味を定義した。そして、定義したものが意図通りの動きを実現しているかどうかをリダクションを行なって動作の確認を行なった。

1.4 本稿の構成

本稿の構成は以下の通りである。

第 2 章では、移動エージェントについてその特徴と実装言語について述べる。

第 3 章では、形式的意味論の説明、意義、および必要性について述べる。

第 4 章では、本稿で用いられる Ambient Calculus について詳しく述べる。

第 5 章では、移動エージェントの切断時操作、フォールトトレランスの定義について述べる。

第 6 章では、第 5 章において定義されたものにおける動作の確認について述べる。

第 7 章は、本稿の結論である。

第 2 章

移動エージェント

移動エージェントとは、データを伴いながらネットワーク上を移動し実行されるプログラムであり、遠隔プロセスで送付先の代理処理を行なうエージェントである（図 2.1）。

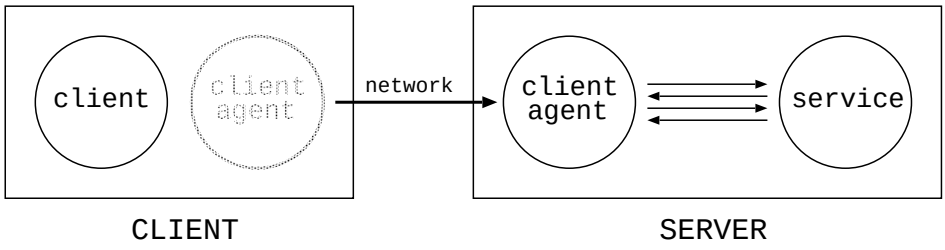

図 2.1: 移動エージェント

移動エージェントにはさまざまな特徴が存在し、またこれを実装する言語も開発されてきている。

本章では、この移動エージェントについて説明を行なう。

2.1 移動エージェントの特徴

移動エージェントには、移動性、自律性、コミュニケーション、並列性、セキュリティなど、求められるものが多々ある。

- 移動性　　ネットワーク内をあるサーバから別のサーバに移動して、様々なサーバ上で処理を行なうこと。

- 自律性　　自己の行動や内部状態を制御する問題解決機構を持ち、他のエージェントや人間からの干渉を受けることなく自律的に動作すること。
- コミュニケーション　さまざまな異なるエージェントや人間と対話することによって必要な情報を集めること。
- 並列性　　各プロセスが独立に動作すること。
- セキュリティ　　移動エージェントをデータの傍受や改ざんから守ること。

これらの特徴を持った移動エージェントは、移動することにより通信の常時接続を前提としないこと、クライアント側の通信環境の悪化に対応できること、通信のセキュリティーが低コストで実現できること、ネットワーク上でのエージェント同士の情報交換のフットワークが軽いことなどのメリットがあり、移動エージェントに関する研究が現在も意欲的にすすめられている。

2.2 移動エージェント言語

移動エージェント言語がさまざまな視野から開発されており、それぞれ異なる特徴を持つ。それらの移動エージェント言語を切断時操作、フォールトトレランスなどの側面から見る。

Kafka(富士通研究所 : Beta 1.3 97/06/24) では、クラス変更に伴うフォールト・トレランスは保証されるが、クラスの意味的な相違に基づく矛盾への対応はエージェントでは関知していない [5]。

また、Aglets（IBM 東京基礎研究所 : ASDK V1.0.2 98/07/05）では、常駐監視作業、非同期作業などにより、断続的な接続によるネットワークの利用が可能になっている [6]。

これらの様に、移動エージェント言語は実用に耐えるアプリケーションを提供するための研究が行なわれている。しかし、装置故障やトラブルが発生した場合の回復機能を備えたものはまだ難しく、これからの研究課題となっている。

2.3 移動エージェントを表現する calculus

移動エージェントを表現するには、2.1 で示した特徴を表すことができなければならない。

コミュニケーションにおいては、メッセージ通信による計算に関して理論的な考察をするための数学的な枠組である CCS(Calculus of Communicating Systems)[3] が存在する。CCS のアクションには、相補的なアクションがあり、それによってエージェント間のコミュニケーションがとられる。

さらにこれを発展させた π-calculus[4] が存在する。これは CCS に通信ポート名の受け渡し機構を拡張したものであり、プロセスへ照合するチャンネルの移動によってプロセスの移動を表すものである。

そして、この π-calculus の概念と手法を基礎とした Ambient Calculus[2] は、計算が生じている境界域である ambient ごと移動を生じさせる。その手法が移動エージェントを表現するのに適していると考え、これを元に研究を行なった。

第 3 章

形式的意味論

プログラミング言語には、構文的な性質と意味的な性質がある。構文的な性質とは、その言語のなかで許されるプログラムを規定するものであり、意味的な性質とは、プログラムの効果及び実行によって得られる結果である。多くのプログラミング言語の仕様には、BNF 記法などによる形式的な構文定義が示されているが、その構文に対応するプログラミング言語の意味については、自然言語によって非形式的に解説している場合が少なくない。このような意味に対する曖昧性を排除し、プログラム言語の意味を数学的あるいは形式論理的な体系によって厳密に記述する方法がプログラミング言語の形式的意味論である。

形式的意味記述法には代表的なものとして、操作的意味論、表示的意味論、公理的意味論がある。操作的意味論とは、プログラム言語の効果を判定できる仕組みをもつ抽象機械を与えることによって、プログラミング言語の意味を表現するものである。表示的意味論とは、プログラム要素をすべて数学的な表示に関連付けることによって、プログラミング言語の意味を表現するものである。公理的意味論とは、プログラム要素を公理、定理などの論理式によって正しさ、誤りを立証するものである。

形式的意味論は、プログラム理論の中核を占め、特に以下のような効用がある。

- プログラム言語の標準化、および言語に携わる者の間に言語の厳密な意味の共有を可能にする。
- 計算機に依存しない形での処理系の厳密な仕様の記述を可能にする。
- 処理系の正当性の検証や処理系の自動生成を可能にする。
- プログラムの正当性などの性質の検証を可能にする。

- 言語の意味を具体的に形式的に記述し、その結果、言語設計の改良に役立つ。

操作的意味論は、ある言語で書かれた任意のプログラムの効果を判定できる仕組みを与えることによって、その言語の意味を表現するものである。この仕組みは解釈型オートマトン（interpreting automaton）といい、形式的にプログラムを実行する能力を持つ形式的な装置である。

この手法は、対象となるプログラミング言語について具体的で直観的な記述を提供することができる。そして、そうした記述を実行する仕組みを容易に作成できるため、言語機能をテストし、その効果をシミュレートできる。

このように機能の説明をより詳しく提供できるため、本研究では、形式的意味論のうち操作的意味論を用いた。

第 4 章

Ambient Calculus

本章では、場所や、エージェントを表現できる ambient という概念を紹介している Mobile Ambients に出てくる Ambient Calculus について説明を行なう。[2]

4.1 概念

Ambient Calculus は、**ambient** の概念と ambient に対しての侵入、退出、解放といった特殊なアクションを用い、ファイアウォールへのアクセスや、自然数を表現できる Turing Complete な calculus である。さらに、プロセス間の通信の概念を加えることにより、π-calculus のエンコードまでを可能にし、表現力が高いことを示している。

4.2 定義・規則

Ambient Calculus で用いられる定義・規則を説明していく。

4.2.1 基本要素

移動、及び通信の基本要素は次の様に表される。

$P, Q ::=$	processes	$M ::=$	capabilities
$(\nu n)P$	restriction	x	variable
$\mathbf{0}$	inactivity	n	name
$P \mid Q$	composition	$in\ M$	can enter into M
$!P$	replication	$out\ M$	can exit out of M
$M[P]$	ambient	$open\ M$	can open M
$M.P$	capability action	ε	null
$(x).P$	input action	$M.M'$	pass
$\langle M \rangle$	async output action		

4.2.2 リダクション規則

Ambient Calculus のリダクション規則を説明していく。

リダクションを視覚的にわかりやすくするために、ambient をフォルダにみたてた図を用いて説明する。[1]（図 4.1参照）

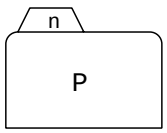

図 4.1: フォルダ（$n[P]$）

侵入 (in)

$in\ m.P$ は、これを囲んでいる ambient へ並列に存在する名前 m の ambient に入るように命令する。もし、並列に存在する名前 m の ambient が存在しないならば、そのような状態になるまで作業を止めることとなる。また、もし１つ以上の並列に存在する名前 m の ambient があるならば、それらの内の１つを選ぶことができる。

$$n[in\ m.P \mid Q] \mid m[R] \rightarrow m[n[P \mid Q] \mid R]$$ （図 4.2参照）

もし、この capability が成功したならば、ambient n は上記のリダクションの様に並列して存在する ambient m 内に入る。その後、$in\ m.P$は P として作業を続ける。

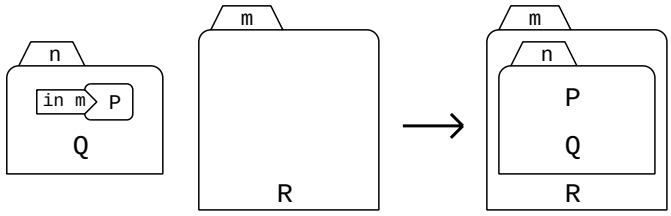

図 4.2: 侵入リダクション

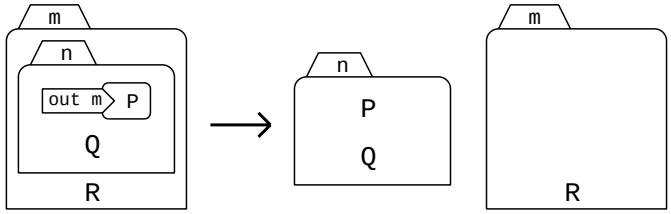

図 4.3: 退出リダクション

退出 (*out*)

out $m.P$ は、これを囲んでいる ambient へその ambient の親にあたる (直接囲んでいる) 名前 m の ambient から出るように命令する。もし、名前 m の ambient が親でなければ、そのような状態になるまで作業を止めることとなる。

$$m[n[out\ m.P \mid Q] \mid R] \rightarrow n[P \mid Q] \mid m[R]$$ （図 4.3参照）

もし、この capability が成功したならば、ambient n は上記のリダクションの様に親である ambient m の外に出て、ambient m と並列になる。その後、*out* $m.P$は P として作業を続ける。

解放 (*open*)

open $n.P$ は、これと並列に存在する名前 n である ambient の境界を分解するように命令する。もし、並列に存在する名前 n の ambient が存在しないならば、そのような状態になるまで作業を止めることとなる。また、もし１つ以上の並列に存在する名前 n の ambient があるならば、それらの内の１つを選ぶことができる。

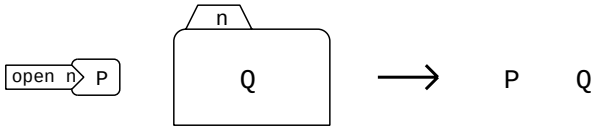

図 4.4: 解放リダクション

$$open\ n.P \mid n[Q] \rightarrow P \mid Q \quad （図 4.4参照）$$

open 命令は、P と Q 双方の上位を狂わせることになるかもしれない。P の視点から見ると、束縛が解かれたとき、Q がどのような動きをするかがわからないのであり、Q の視点から見ると、自分の環境が引き裂かれることとなる。それにもかかわらず、この命令が比較的行儀良く動作するのは、以下の理由による。

- 分解は $open\ n.P$ によって起こるので、P と並列に Q が出現することはまったく予測できないわけではない
- $open\ n$ は、n によって与えられる capability なので、望まなければ、$n[Q]$ を分解させることはできない。

複製 (!)

$!P$ は、π-calculus[4] と同じ仕様であり、望むだけの数の複写を意味する。

$$!P \rightarrow P \mid !P \quad （図 4.5参照）$$

しかし、これで無限並列動作が起こる危険はない。複製の共通の使い方は$!M.P$ であり、ambient M が M を経てコミュニケーションをとったときにのみ複製させることができる。

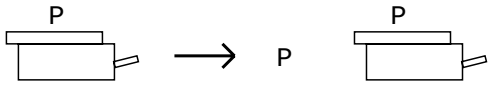

図 4.5: 複製リダクション

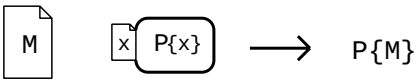

図 4.6: 入出力リダクション

入出力 ($(\,) \bullet \langle\rangle$)

出力は、直接囲んでいる ambient 内でのローカルな入力待ちに capability (あるいは name) を放つ。入力は、ローカルな入力待ちから capability を獲得し、範囲内での変数をバインドする。

$$(x).P \mid \langle M \rangle \to P\{x \leftarrow M\} \quad \text{（図 4.6 参照）}$$

このローカルな通信機構は、ambient によく適応する。特に、遠距離を移動するような遠距離通信は、ファイアウォールを通るため自動的に起こるべきではない。それにもかかわらずこのシンプルな機構は、名前の付いたチャンネルを通して通信をエミュレートし、ごく普通に非同期 π-calculus のエンコードを提供するのに十分である。

4.2.3 推論規則

推論規則には以下のものが存在する。

$$\begin{aligned}
P' \equiv P, P \to Q, Q \equiv Q' \;&\Rightarrow\; P' \to Q' \\
P \to Q \;&\Rightarrow\; (\nu n)P \to (\nu n)Q \\
P \to Q \;&\Rightarrow\; M[P] \to M[Q] \\
P \to Q \;&\Rightarrow\; (x).P \to (x).Q \\
P \to Q \;&\Rightarrow\; P \mid R \to Q \mid R
\end{aligned}$$

4.2.4 略記法

略記法は以下のものを使う。

$$\begin{aligned}(\nu n_1 \ldots \nu n_m)P &\triangleq (\nu n_1)\ldots(\nu n_m)P \\ n[] &\triangleq n[\mathbf{0}] \\ M &\triangleq M.\mathbf{0}\end{aligned}$$

4.2.5 等式

構造的合同なものに以下のものが挙げられる。

$$\begin{array}{ll}
P \equiv P & P \mid Q \equiv Q \mid P \\
P \equiv Q \;\Rightarrow\; Q \equiv P & (P \mid Q) \mid R \equiv P \mid (Q \mid R) \\
P \equiv Q, Q \equiv R \;\Rightarrow\; P \equiv R & (M.M').P \equiv M.M'.P \\
 & (\nu n)(\nu m)P \equiv (\nu m)(\nu n)P \\
P \equiv Q \;\Rightarrow\; (\nu n)P \equiv (\nu n)Q & (\nu n)(P \mid Q) \equiv P \mid (\nu n)Q \quad \text{if } n \notin fn(P) \\
P \equiv Q \;\Rightarrow\; P \mid R \equiv Q \mid R & (\nu n)(m[P]) \equiv m[(\nu n)P] \quad \text{if } n \neq m \\
P \equiv Q \;\Rightarrow\; !P \equiv !Q & \\
P \equiv Q \;\Rightarrow\; M[P] \equiv M[Q] & P \mid \mathbf{0} \equiv P \\
P \equiv Q \;\Rightarrow\; M.P \equiv M.Q & (\nu n)\mathbf{0} \equiv \mathbf{0} \\
P \equiv Q \;\Rightarrow\; (x).P \equiv (x).Q & !\mathbf{0} \equiv \mathbf{0} \\
 & \varepsilon.P \equiv P
\end{array}$$

$fn(P)$ は、P においてフリーである name の集合である。

4.3 Ambient Calculus を用いた例：ロック機構

第 4 章で述べてきた Ambient Calculus を用いた例として、ロック機構を紹介し、これを用いて、2つのエージェントを同期させるモデルを示す。

ロックの機構は以下のように書くことができる。ロックの取得および、解放のモデルは次の通り。

$$\text{acquire } n.P \triangleq \mathit{open}\ n.P \qquad \text{release } n.P \triangleq n[] \mid P$$

これらを使い、2つのエージェントを同期させることができるようになる。

$$\text{acquire } n.\text{release } m.P \mid \text{release } n.\text{acqire } m.Q$$

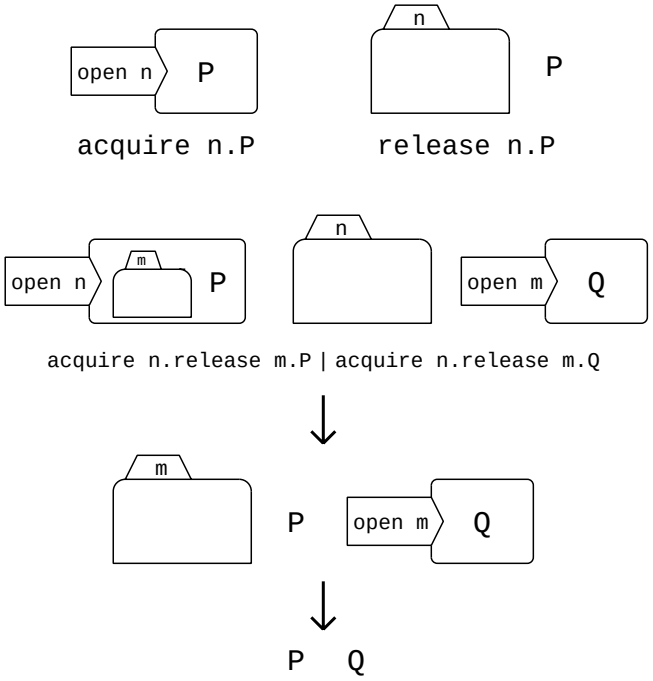

図 4.7: ロック機構

リダクションは以下の通り。なお、アンダーラインは次のリダクション時に働くものに引いてある。

$$\begin{aligned}
&\text{acquire } n.\text{release } m.P \mid \text{release } n.\text{acqire } m.Q \\
&\triangleq \underline{open\ n}.(m[] \mid P) \mid \underline{n}[] \mid open\ m.Q \\
&\rightarrow \underline{m}[] \mid P \mid \underline{open\ m}.Q \\
&\rightarrow P \mid Q
\end{aligned}$$

フォルダを用いた説明を、図 4.7に示す。

第 5 章

移動エージェントの表現

Ambient Calculus[2] は、移動性、自律性、コミュニケーション、並列性を表現でき、ファイアウォールなどのセキュリティについてのモデルも表現できる能力を持つ。

この章は、本研究の骨子である、通信の切断があった場合の移動エージェントの動作の意味を Ambient Calculus によって定義していく。

5.1 共通に用いる表現

5.1.1 カウントダウン

ambient p に直接囲まれている場合に、i から 0 までをカウントダウンしていく capability を count $p\ i$ と表記する。

カウントが 0 になった時は、capability は null となる。

$$p[count\ p\ 0.P] \equiv p[P]$$

カウントダウンにおけるリダクションの表記を、通常のリダクションに用いる$\longrightarrow$ の上に、past $p\ i$ と書くことにより表すものとする。$\overset{past\ p\ i}{\longrightarrow}$は、ambient p 内でカウントが i 下がることを意味する。

$$p[count\ p\ i.P] \overset{past\ p\ 1}{\longrightarrow} p[count\ p\ (i-1).P]$$

カウントダウンする時間の単位は現在未定義であるが、ambient 内でのローカルな値とする。つまり、ambient p 内でカウントが 1 下がったとしても、他の ambient 内でのカウントも同じく 1 下がるとは限らない。つまり、

$$p[count\ p\ i.P] \mid q[count\ q\ j.Q] \stackrel{past\ p}{\longrightarrow}{}^{1} p[count\ p\ (i-1).P] \mid q[count\ q\ (j-1).Q]$$

となるとは限らないのである。

5.1.2 動作不能状態

name で束縛された ambient 内に、その束縛された name を持つ capability が存在しない状態にし、ambient 内のプロセスを動作不能状態にするものを考える。

ambient p に対して動作不能状態を起こすものを、ambient p を封じ込める ambient l と、ambient p に侵入し ambinet p を ambinet l につれてくる ambient k とを組み合わせて作る。

$$\text{lock } p \triangleq (\nu\ l)(l[] \mid (\nu\ k)k[in\ p.key[out\ k.open\ k.in\ l]])$$

また、動作不能状態にされる側の ambient も、そのような状態になることを許すプロセスを置く。これがなければ、lock によって動作不能状態になることはない。

$$\text{permit lock} \triangleq open\ key$$

実際にこれらを作用させると以下のようになる。

$$\text{lock } p \mid p[P \mid \text{permit lock}] \rightarrow^{*} (\nu\ l)l[p[P]]$$

これにより、ambient l は内部からも外部からも操作できなくなる。

動作の詳細は後述する。

5.2 切断時操作の定義

ここで定義される切断時操作とは、常に作業状況を返しながら作業をつづけるものにおいて考える。

例としては、実行中の移動エージェントが現在どのぐらいの作業を終了しているのかを送信元に表示する場合などを想定している。そして通信路が切断されている場合は表示用データを送信できないために送信を止め、通信路が切断解除されたときにその時の最新情報のみを送信するものとする。(図 5.1)

切断時操作は、

- 通信が切断されていない場合は通常通り動作を行ない、

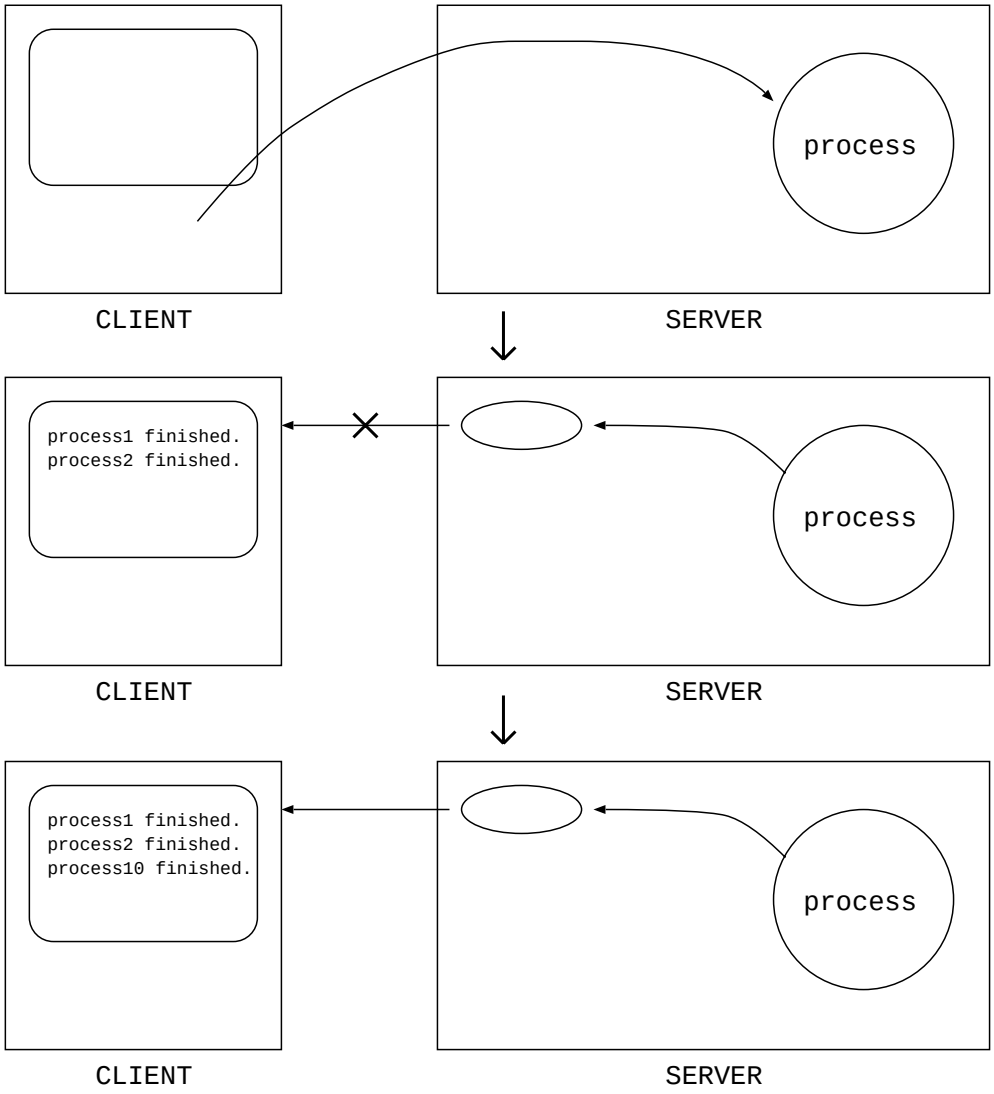

図 5.1: 作業状況を返す場合の切断時操作

- 通信が切断されていた場合でも通常通り動作を行ない、
- 結果などを送信するときに通信が切断されていた場合は、通信が再開されるまで待つ

ものと考えられる。

そのため、通信状態の確認を行なうものを考え、定義した。

以下に、その定義を示していく。

5.2.1 ambient 間ルート確認

ambient 間の通信が切断されていないかどうかを確認するためのプロセスを考える。

これは、送信用のデータが存在することを確認したのち ambient 間を往復し、通信が切断されていないことが確認されたら、データを送信するための情報を渡すものを考える。

$$\text{verify route from } b \text{ to } a \triangleq !open\ p.r[out\ b.in\ a.out\ a.in\ b.in\ s.\langle out\ b.in\ a\rangle]$$

5.2.2 データ送信用ベース

送信するデータを受けとったのち、通信が切断されていないという情報を受けとるとデータを送信し、一定時間内に通信に関する情報が受けとれない場合はその情報を消してしまい、新たな情報を待つものを考える。

$$\text{trans base} \triangleq !(y).s[(x).q[out\ s.x.\langle y\rangle] \mid \text{permit lock} \mid open\ r]$$

x と y、2つの入力待ちはそれぞれ、移動先指定データ、送信用データを受けとる部分であり、送信用データは ambient q に伴われて移動する。

5.2.3 逐次データ配送用ベース

データが送信されたか消されたのちに、次のデータを配送する機能を持つものを考える。この概念の機能は細かく分けると3つに分類される。それは、

- 送信用データを受け入れるための応答
- 送信用データ格納
- 逐次にデータを配送するための区切り

である。これらを用いて、逐次データ配送用ベースを考える。

$$\text{data delivery from } b \triangleq$$
$$t[!open\ u \mid !(x).p[out\ t.\langle x\rangle \mid \text{count } b\ i.(\text{lock } s \mid u[open\ v])]] \mid u[open\ v]$$

5.2.4 送信データ

送信するデータを考える。

$$\text{data } M \triangleq v[in\ u.in\ t.\langle M \rangle]$$

$in\ u$ は前に送り出したデータが送信されたか、消去されたかした場合に働き、その後 $in\ t$ によって、逐次データ配送用ベースへと送りこまれることとなる。

5.2.5 切断時操作

ambient 間ルート確認、データ送信用ベース、逐次データ配送用ベース、送信データを用いて、常に途中結果を返しながら作業をつづけ、送信先との通信が切断されている場合は、送ろうとしていたデータを破棄し、通信が再開された場合、その時の最も新しいデータから送信し始めるものについて考える。(図 5.2)

$$a[Q] \mid b[\text{verify route from } b \text{ to } a \mid \text{trans base} \mid \text{data delivery from } b \mid \text{data } M \mid P]$$

5.3 物理的な切断が起こった場合のフォールトトレランスの定義

２つの ambient 間の通信状態を常に確かめ、もし一定時間内に通信が行なえなかった場合は用意しておいた非常用プロセスを走らせるというものを考えていく。これにより、物理的な切断が起こった場合のフォールトトレランスの定義を行なう。

フォールトトレランスとは、システムの一部に何らかの障害が発生した場合でも，システムを停止せずに継続処理できるようにすることである。

この研究では、ホストに結果を返すべきエージェントが存在するサーバとの通信が永続的に切断された場合のフォールトトレランスを想定している。

そのため、通信の切断が起こってないかどうかを常に確かめておく必要があると考え、プロセス間を常に往復するエージェントと、それを関知するエージェントを定義し、これを元にフォールトトレランスを行なうものを定義した。

以下に、その定義を示していく。

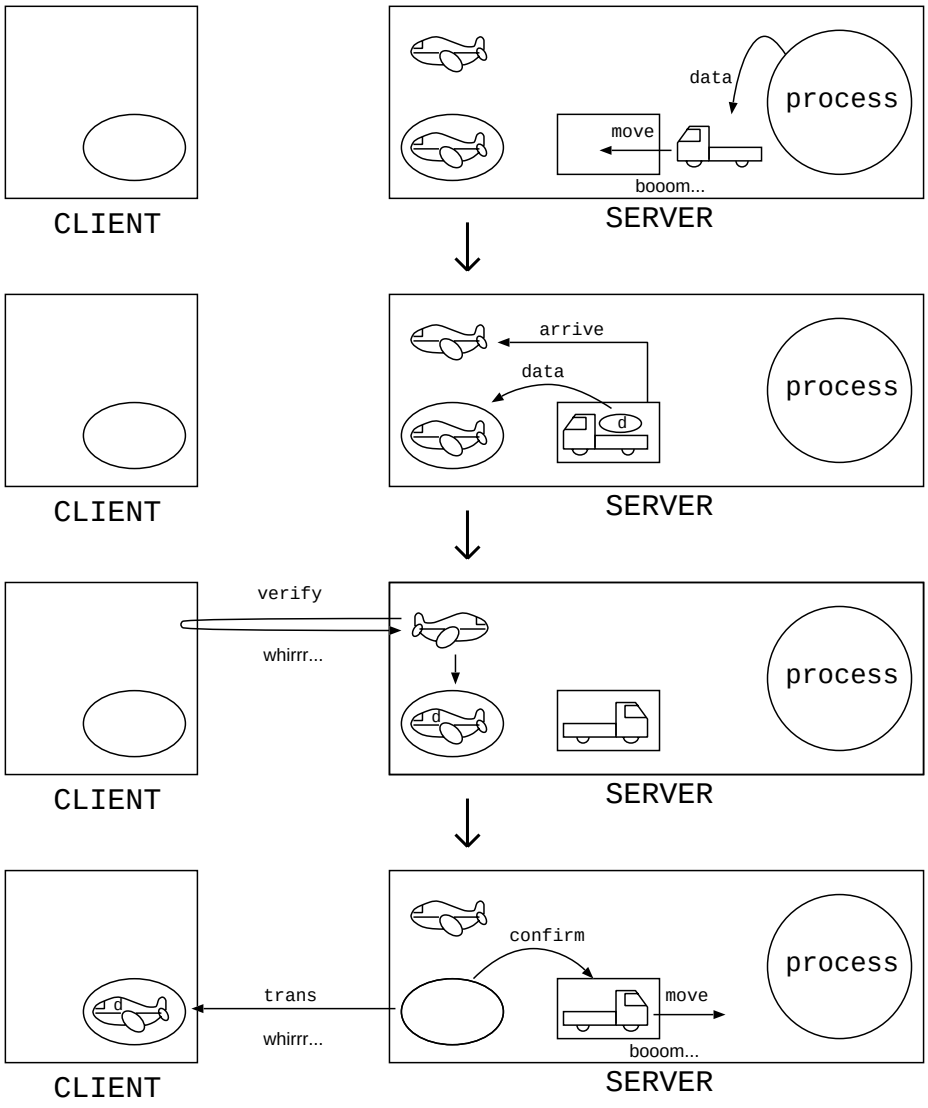
data
process
move
booom...
CLIENT
SERVER
arrive
data
process
d
CLIENT
SERVER
verify
whirrr...
process
d
CLIENT
SERVER
confirm
process
trans
move
whirrr...
booom...
d
CLIENT
SERVER

図 5.2: 切断時操作

5.3.1 往復エージェント用ベース

往復エージェントが戻ってきた時に再び往復できる状態に戻すためのプロセスを考える。

往復エージェント用ベースは永続的に作用する状態にしなければならないので個々のプロセスは複製を用い、そのプロセスの種類は３つに分けて考える。それは、

- 往復エージェントが戻ってきたとの通信を受け、再び動き出すための capability などを作りだすプロセス、
- 一定時間内に戻らなかった時に動き出す非常用プロセス、
- 非常用プロセスを停止させるプロセスを作動させるための鍵となるプロセス

である。この３つを組み合わせて、往復エージェント用ベースを考える。

$$\begin{aligned}\text{base-set} \triangleq\ & !(x).l[in\ m.x.n[\langle x\rangle \mid out\ m.q[out\ n.\text{lock}\ p]]] \\ & \mid\ !open\ n.p[\text{count}\ p\ i.P \mid \text{permit lock}] \mid !open\ q\end{aligned}$$

5.3.2 往復エージェント

実際に往復するエージェントを考える。

これには移動の核となる ambient と、移動・到着などの情報を積んだ ambient を組み合わせることとなる。

$$\text{round-agent}\ \ \text{from}\ \ a\ \ \text{to}\ \ b \triangleq m[!open\ l] \mid n[\langle out\ a.in\ b.out\ b.in\ a\rangle]$$

ambient m 内の $!openl$ は、移動する情報を積んだあとに着火するための capability である。また、ambient n 内の出力プロセスは、ベースに送ったのち移動するための情報を得ることになる。

5.3.3 ambient 間連続往復エージェント

往復エージェント用ベースと往復エージェントを用い、通信が続いている限り常に同じ状態に戻り続ける ambient 間連続往復エージェントを考える。

$$a[\text{base-set} \mid \text{round-agent}\ \ \text{from}\ \ a\ \ \text{to}\ \ b] \mid b[Q]$$

これは、ambient a と ambient b とを往復エージェントが走り、再び上記と全く同じ状態に戻ることとなる。

動作の詳細は後述する。

5.3.4 物理的な切断が起こった場合のフォールトトレランス

ambient 間連続往復エージェントが、一定時間を経過してもベースに戻ってこないときに非常用プロセスを作動させ、その処理においてフォールトトレランスを行なうものとする。

第 6 章

動作の確認

動作不能状態、切断時操作、ambient 間連続往復エージェントの動作を確認し、仕様通りの動作を行なうことを確かめていく。

6.1 動作不能状態のリダクション

アンダーラインは次のリダクション時に働くものに引いてある。

$$
\begin{aligned}
&\text{lock } p \mid p[P \mid \text{permit lock}] \\
&\triangleq (\nu\ l)(l[] \mid (\nu\ k)k[\underline{in\ p}.key[out\ k.open\ k.in\ l]]) \mid \underline{p}[P \mid open\ key] \\
&\rightarrow (\nu\ l')(l'[] \mid p[P \mid open\ key \mid (\nu\ k)\underline{k}[key[\underline{out\ k}.open\ k.in\ l']]]) \quad (l' \notin fn(P)) \\
&\rightarrow (\nu\ l')(l'[] \mid p[P \mid \underline{open\ key} \mid (\nu\ k)(\underline{key}[open\ k.in\ l'] \mid k[])]) \\
&\rightarrow (\nu\ l')(l'[] \mid p[P \mid (\nu\ k)(\underline{open\ k}.in\ l' \mid \underline{k}[])]) \\
&\rightarrow (\nu\ l')(\underline{l'}[] \mid p[P \mid \underline{in\ l'}]) \\
&\rightarrow (\nu\ l')l'[p[P]]
\end{aligned}
$$

6.2 切断時操作のリダクション

アンダーラインは次のリダクション時に働くものに引いてある。

$$
\begin{aligned}
&a[Q] \mid b[\text{verify route from } a \text{ to } b \mid \text{trans base} \mid \text{data delivery from } b \mid \text{data } M \mid P] \\
&\triangleq a[Q] \mid b[!open\ p.r[out\ b.in\ a.out\ a.in\ b.in\ s.\langle out\ b.in\ a\rangle] \mid !(y).s[(x).q[out\ s.x.\langle y\rangle] \\
&\qquad \mid \text{permit lock} \mid open\ r] \mid t[!open\ u \mid !(x).p[out\ t.\langle x\rangle \mid \text{count } b\ i.(\text{lock } s \mid u[open\ v])]]
\end{aligned}
$$

$$
\begin{aligned}
&\quad \mid \underline{u}[\mathit{open}\ v] \mid v[\underline{\mathit{in}\ u}.\mathit{in}\ t.\langle M\rangle] \mid P] \\
&\to a[Q] \mid b[!\mathit{open}\ p.r[\mathit{out}\ b.\mathit{in}\ a.\mathit{out}\ a.\mathit{in}\ b.\mathit{in}\ s.\langle \mathit{out}\ b.\mathit{in}\ a\rangle] \mid !(y).s[(x).q[\mathit{out}\ s.x.\langle y\rangle] \\
&\quad \mid \text{permit lock} \mid \mathit{open}\ r] \mid t[!\mathit{open}\ u \mid !(x).p[\mathit{out}\ t.\langle x\rangle \mid \text{count}\ b\ i.(\text{lock}\ s \mid u[\mathit{open}\ v])]] \\
&\quad \mid u[\underline{\mathit{open}\ v} \mid \underline{v}[\mathit{in}\ t.\langle M\rangle]] \mid P] \\
&\to a[Q] \mid b[!\mathit{open}\ p.r[\mathit{out}\ b.\mathit{in}\ a.\mathit{out}\ a.\mathit{in}\ b.\mathit{in}\ s.\langle \mathit{out}\ b.\mathit{in}\ a\rangle] \mid !(y).s[(x).q[\mathit{out}\ s.x.\langle y\rangle] \\
&\quad \mid \text{permit lock} \mid \mathit{open}\ r] \mid \underline{t}[!\mathit{open}\ u \mid !(x).p[\mathit{out}\ t.\langle x\rangle \mid \text{count}\ b\ i.(\text{lock}\ s \mid u[\mathit{open}\ v])]] \\
&\quad \mid u[\underline{\mathit{in}\ t}.\langle M\rangle] \mid P] \\
&\to a[Q] \mid b[!\mathit{open}\ p.r[\mathit{out}\ b.\mathit{in}\ a.\mathit{out}\ a.\mathit{in}\ b.\mathit{in}\ s.\langle \mathit{out}\ b.\mathit{in}\ a\rangle] \mid !(y).s[(x).q[\mathit{out}\ s.x.\langle y\rangle] \\
&\quad \mid \text{permit lock} \mid \mathit{open}\ r] \mid t[!\underline{\mathit{open}\ u} \mid !(x).p[\mathit{out}\ t.\langle x\rangle \mid \text{count}\ b\ i.(\text{lock}\ s \mid u[\mathit{open}\ v])] \mid \underline{u}[\langle M\rangle]] \\
&\quad \mid P] \\
&\to a[Q] \mid b[!\mathit{open}\ p.r[\mathit{out}\ b.\mathit{in}\ a.\mathit{out}\ a.\mathit{in}\ b.\mathit{in}\ s.\langle \mathit{out}\ b.\mathit{in}\ a\rangle] \mid !(y).s[(x).q[\mathit{out}\ s.x.\langle y\rangle] \\
&\quad \mid \text{permit lock} \mid \mathit{open}\ r] \mid t[!\mathit{open}\ u \mid !\underline{(x)}.p[\mathit{out}\ t.\langle x\rangle \mid \text{count}\ b\ i.(\text{lock}\ s \mid u[\mathit{open}\ v])] \mid \underline{\langle M\rangle}] \\
&\quad \mid P] \\
&\to a[Q] \mid b[!\mathit{open}\ p.r[\mathit{out}\ b.\mathit{in}\ a.\mathit{out}\ a.\mathit{in}\ b.\mathit{in}\ s.\langle \mathit{out}\ b.\mathit{in}\ a\rangle] \mid !(y).s[(x).q[\mathit{out}\ s.x.\langle y\rangle] \\
&\quad \mid \text{permit lock} \mid \mathit{open}\ r] \mid \underline{t}[!\mathit{open}\ u \mid !(x).p[\mathit{out}\ t.\langle x\rangle \mid \text{count}\ b\ i.(\text{lock}\ s \mid u[\mathit{open}\ v])] \\
&\quad \mid p[\underline{\mathit{out}\ t}.\langle M\rangle \mid \text{count}\ b\ i.(\text{lock}\ s \mid u[\mathit{open}\ v])]] \mid P] \\
&\to a[Q] \mid b[!\underline{\mathit{open}\ p}.r[\mathit{out}\ b.\mathit{in}\ a.\mathit{out}\ a.\mathit{in}\ b.\mathit{in}\ s.\langle \mathit{out}\ b.\mathit{in}\ a\rangle] \mid !(y).s[(x).q[\mathit{out}\ s.x.\langle y\rangle] \\
&\quad \mid \text{permit lock} \mid \mathit{open}\ r] \mid t[!\mathit{open}\ u \mid !(x).p[\mathit{out}\ t.\langle x\rangle \mid \text{count}\ b\ i.(\text{lock}\ s \mid u[\mathit{open}\ v])]] \\
&\quad \mid \underline{p}[\langle M\rangle \mid \text{count}\ b\ i.(\text{lock}\ s \mid u[\mathit{open}\ v])] \mid P] \\
&\to \underline{a}[Q] \mid \underline{b}[!\mathit{open}\ p.r[\mathit{out}\ b.\mathit{in}\ a.\mathit{out}\ a.\mathit{in}\ b.\mathit{in}\ s.\langle \mathit{out}\ b.\mathit{in}\ a\rangle] \mid !\underline{\underline{(y)}}.s[(x).q[\mathit{out}\ s.x.\langle y\rangle] \\
&\quad \mid \text{permit lock} \mid \mathit{open}\ r] \mid t[!\mathit{open}\ u \mid !(x).p[\mathit{out}\ t.\langle x\rangle \mid \text{count}\ b\ i.(\text{lock}\ s \mid u[\mathit{open}\ v])]] \\
&\quad \mid r[\underline{\mathit{out}\ b.\mathit{in}\ a}.\mathit{out}\ a.\mathit{in}\ b.\mathit{in}\ s.\langle \mathit{out}\ b.\mathit{in}\ a\rangle] \mid \underline{\underline{\langle M\rangle}} \mid \text{count}\ b\ i.(\text{lock}\ s \mid u[\mathit{open}\ v]) \mid P] \\
&\to \underline{a}[r[\underline{\mathit{out}\ a.\mathit{in}\ b}.\mathit{in}\ s.\langle \mathit{out}\ b.\mathit{in}\ a\rangle] \mid Q] \mid \underline{b}[!\mathit{open}\ p.r[\mathit{out}\ b.\mathit{in}\ a.\mathit{out}\ a.\mathit{in}\ b.\mathit{in}\ s.\langle \mathit{out}\ b.\mathit{in}\ a\rangle] \\
&\quad \mid !(y).s[(x).q[\mathit{out}\ s.x.\langle y\rangle] \mid \text{permit lock} \mid \mathit{open}\ r] \mid t[!\mathit{open}\ u \mid !(x).p[\mathit{out}\ t.\langle x\rangle \\
&\quad \mid \text{count}\ b\ i.(\text{lock}\ s \mid u[\mathit{open}\ v])]] \mid s[(x).q[\mathit{out}\ s.x.\langle M\rangle] \mid \text{permit lock} \mid \mathit{open}\ r] \\
&\quad \mid \text{count}\ b\ i'.(\text{lock}\ s \mid u[\mathit{open}\ v]) \mid P] \\
&\to a[Q] \mid b[!\mathit{open}\ p.r[\mathit{out}\ b.\mathit{in}\ a.\mathit{out}\ a.\mathit{in}\ b.\mathit{in}\ s.\langle \mathit{out}\ b.\mathit{in}\ a\rangle] \mid !(y).s[(x).q[\mathit{out}\ s.x.\langle y\rangle] \\
&\quad \mid \text{permit lock} \mid \mathit{open}\ r] \mid t[!\mathit{open}\ u \mid !(x).p[\mathit{out}\ t.\langle x\rangle \mid \text{count}\ b\ i.(\text{lock}\ s \mid u[\mathit{open}\ v])]] \\
&\quad \mid r[\underline{\mathit{in}\ s}.\langle \mathit{out}\ b.\mathit{in}\ a\rangle] \mid \underline{s}[(x).q[\mathit{out}\ s.x.\langle M\rangle] \mid \text{permit lock} \mid \mathit{open}\ r] \\
&\quad \mid \text{count}\ b\ i'.(\text{lock}\ s \mid u[\mathit{open}\ v]) \mid P] \\
&\to a[Q] \mid b[!\mathit{open}\ p.r[\mathit{out}\ b.\mathit{in}\ a.\mathit{out}\ a.\mathit{in}\ b.\mathit{in}\ s.\langle \mathit{out}\ b.\mathit{in}\ a\rangle] \mid !(y).s[(x).q[\mathit{out}\ s.x.\langle y\rangle] \\
&\quad \mid \text{permit lock} \mid \mathit{open}\ r] \mid t[!\mathit{open}\ u \mid !(x).p[\mathit{out}\ t.\langle x\rangle \mid \text{count}\ b\ i.(\text{lock}\ s \mid u[\mathit{open}\ v])]] \\
&\quad \mid s[\underline{r}[\langle \mathit{out}\ b.\mathit{in}\ a\rangle] \mid (x).q[\mathit{out}\ s.x.\langle M\rangle] \mid \text{permit lock} \mid \underline{\mathit{open}\ r}]
\end{aligned}
$$

$$
\begin{aligned}
&\quad \mid \text{count } b\ i'.(\text{lock } s \mid u[open\ v]) \mid P] \\
&\to a[Q] \mid b[!open\ p.r[out\ b.in\ a.out\ a.in\ b.in\ s.\langle out\ b.in\ a\rangle] \mid !(y).s[(x).q[out\ s.x.\langle y\rangle] \\
&\quad \mid \text{permit lock} \mid open\ r] \mid t[!open\ u \mid !(x).p[out\ t.\langle x\rangle \mid \text{count } b\ i.(\text{lock } s \mid u[open\ v])]] \\
&\quad \mid s[\underline{\langle out\ b.in\ a\rangle} \mid \underline{(x)}.q[out\ s.x.\langle M\rangle] \mid \text{permit lock}] \mid \text{count } b\ i'.(\text{lock } s \mid u[open\ v]) \mid P] \\
&\to a[Q] \mid b[!open\ p.r[out\ b.in\ a.out\ a.in\ b.in\ s.\langle out\ b.in\ a\rangle] \mid !(y).s[(x).q[out\ s.x.\langle y\rangle] \\
&\quad \mid \text{permit lock} \mid open\ r] \mid t[!open\ u \mid !(x).p[out\ t.\langle x\rangle \mid \text{count } b\ i.(\text{lock } s \mid u[open\ v])]] \\
&\quad \mid \underline{s}[q[\underline{out\ s}.out\ b.in\ a.\langle M\rangle] \mid \text{permit lock}] \mid \text{count } b\ i'.(\text{lock } s \mid u[open\ v]) \mid P] \\
&\to \underline{a}[Q] \mid \underline{b}[!open\ p.r[out\ b.in\ a.out\ a.in\ b.in\ s.\langle out\ b.in\ a\rangle] \mid !(y).s[(x).q[out\ s.x.\langle y\rangle] \\
&\quad \mid \text{permit lock} \mid open\ r] \mid t[!open\ u \mid !(x).p[out\ t.\langle x\rangle \mid \text{count } b\ i.(\text{lock } s \mid u[open\ v])]] \\
&\quad \mid q[\underline{out\ b.in\ a}.\langle M\rangle] \mid s[\text{permit lock}] \mid \text{count } b\ i'.(\text{lock } s \mid u[open\ v]) \mid P] \\
&\to a[q[\langle M\rangle] \mid Q] \mid b[!open\ p.r[out\ b.in\ a.out\ a.in\ b.in\ s.\langle out\ b.in\ a\rangle] \mid !(y).s[(x).q[out\ s.x.\langle y\rangle] \\
&\quad \mid \text{permit lock} \mid open\ r] \mid t[!open\ u \mid !(x).p[out\ t.\langle x\rangle \mid \text{count } b\ i.(\text{lock } s \mid u[open\ v])]] \\
&\quad \mid s[\text{permit lock}] \mid \text{count } b\ i'.(\text{lock } s \mid u[open\ v]) \mid P]
\end{aligned}
$$

count $b\ i'.(\text{lock } s \mid u[open\ v])$ において、$i' = 0$ までカウントされたときに、データが送信される前ならば、 ambient s ごとデータをロックし、データが送信された後ならば、データが送信された後の ambient s をロックする。この後、再び P から $v[in\ u.in\ t.\langle M\rangle]$ の形をしたデータが送られてきたら、再び同様のリダクションを行なう。

$$
\begin{aligned}
&a[q[\langle M\rangle] \mid Q] \mid b[!open\ p.r[out\ b.in\ a.out\ a.in\ b.in\ s.\langle out\ b.in\ a\rangle] \mid !(y).s[(x).q[out\ s.x.\langle y\rangle] \\
&\quad \mid \text{permit lock} \mid open\ r] \mid t[!open\ u \mid !(x).p[out\ t.\langle x\rangle \mid \text{count } b\ i.(\text{lock } s \mid u[open\ v])]] \\
&\quad \mid s[\text{permit lock}] \mid \text{count } b\ j.(\text{lock } s \mid u[open\ v]) \mid P] \\
&\xrightarrow{past\ b\ j} a[q[\langle M\rangle] \mid Q] \mid b[!open\ p.r[out\ b.in\ a.out\ a.in\ b.in\ s.\langle out\ b.in\ a\rangle] \mid !(y).s[(x).q[out\ s.x.\langle y\rangle] \\
&\quad \mid \text{permit lock} \mid open\ r] \mid t[!open\ u \mid !(x).p[out\ t.\langle x\rangle \mid \text{count } b\ i.(\text{lock } s \mid u[open\ v])]] \\
&\quad \mid \underline{s[\text{permit lock}]} \mid \underline{\text{lock } s} \mid u[open\ v] \mid P] \\
&\to a[q[\langle M\rangle] \mid Q] \mid b[!open\ p.r[out\ b.in\ a.out\ a.in\ b.in\ s.\langle out\ b.in\ a\rangle] \mid !(y).s[(x).q[out\ s.x.\langle y\rangle] \\
&\quad \mid \text{permit lock} \mid open\ r] \mid t[!open\ u \mid !(x).p[out\ t.\langle x\rangle \mid \text{count } b\ i.(\text{lock } s \mid u[open\ v])]] \\
&\quad \mid u[open\ v] \mid P]
\end{aligned}
$$

6.3 ambient 間連続往復エージェントのリダクション

アンダーラインは次のリダクション時に働くものに引いてある。

a[base-set | round-agent from a to b] | $b[Q]$

$\triangleq a[!(x).l[in\ m.x.n[\langle x\rangle \mid out\ m.q[out\ n.\text{lock}\ p]]] \mid \underline{!open\ n.p}[\text{count}\ p\ i.P \mid \text{permit lock}]$
$\mid !open\ q \mid m[!open\ l] \mid \underline{n}[\langle out\ a.in\ b.out\ b.in\ a\rangle]] \mid b[Q]$

$\rightarrow a[!\underline{(x)}.l[in\ m.x.n[\langle x\rangle \mid out\ m.q[out\ n.\text{lock}\ p]]] \mid !open\ n.p[\text{count}\ p\ i.P$
$\mid \text{permit lock}] \mid !open\ q \mid p[\text{count}\ p\ i.P \mid \text{permit lock}] \mid m[!open\ l]$
$\mid \underline{\langle out\ a.in\ b.out\ b.in\ a\rangle}] \mid b[Q]$

$\rightarrow a[!(x).l[in\ m.x.n[\langle x\rangle \mid out\ m.q[out\ n.\text{lock}\ p]]] \mid !open\ n.p[\text{count}\ p\ i.P$
$\mid \text{permit lock}] \mid !open\ q \mid p[\text{count}\ p\ i'.P \mid \text{permit lock}]$
$\mid l[\underline{in\ m}.out\ a.in\ b.out\ b.in\ a.n[\langle out\ a.in\ b.out\ b.in\ a\rangle \mid out\ m.q[out\ n.\text{lock}\ p]]]$
$\mid \underline{m}[!open\ l]] \mid b[Q]$

$\rightarrow a[!(x).l[in\ m.x.n[\langle x\rangle \mid out\ m.q[out\ n.\text{lock}\ p]]] \mid !open\ n.p[\text{count}\ p\ i.P$
$\mid \text{permit lock}] \mid !open\ q \mid p[\text{count}\ p\ i'.P \mid \text{permit lock}]$
$\mid m[\underline{!open\ l} \mid \underline{l}[out\ a.in\ b.out\ b.in\ a.n[\langle out\ a.in\ b.out\ b.in\ a\rangle \mid out\ m.q[out\ n.\text{lock}\ p]]]]$
$\mid b[Q]$

$\rightarrow \underline{a}[!(x).l[in\ m.x.n[\langle x\rangle \mid out\ m.q[out\ n.\text{lock}\ p]]] \mid !open\ n.p[\text{count}\ p\ i.P$
$\mid \text{permit lock}] \mid !open\ q \mid p[\text{count}\ p\ i'.P \mid \text{permit lock}]$
$\mid m[!open\ l \mid \underline{out\ a.in\ b}.out\ b.in\ a.n[\langle out\ a.in\ b.out\ b.in\ a\rangle \mid out\ m.q[out\ n.\text{lock}\ p]]]] \mid \underline{b}[Q]$

$\rightarrow \underline{a}[!(x).l[in\ m.x.n[\langle x\rangle \mid out\ m.q[out\ n.\text{lock}\ p]]] \mid !open\ n.p[\text{count}\ p\ i.P$
$\mid \text{permit lock}] \mid !open\ q \mid p[\text{count}\ p\ i'.P \mid \text{permit lock}]]$
$\mid \underline{b}[m[!open\ l \mid \underline{out\ b.in\ a}.n[\langle out\ a.in\ b.out\ b.in\ a\rangle \mid out\ m.q[out\ n.\text{lock}\ p]]] \mid Q]$

$\rightarrow a[!(x).l[in\ m.x.n[\langle x\rangle \mid out\ m.q[out\ n.\text{lock}\ p]]] \mid !open\ n.p[\text{count}\ p\ i.P$
$\mid \text{permit lock}] \mid !open\ q \mid p[\text{count}\ p\ i'.P \mid \text{permit lock}]$
$\mid \underline{m}[!open\ l \mid n[\langle out\ a.in\ b.out\ b.in\ a\rangle \mid \underline{out\ m}.q[out\ n.\text{lock}\ p]]]] \mid b[Q]$

$\rightarrow a[!(x).l[in\ m.x.n[\langle x\rangle \mid out\ m.q[out\ n.\text{lock}\ p]]] \mid !open\ n.p[\text{count}\ p\ i.P$
$\mid \text{permit lock}] \mid !open\ q \mid p[\text{count}\ p\ i'.P \mid \text{permit lock}]$
$\mid m[!open\ l] \mid \underline{n}[\langle out\ a.in\ b.out\ b.in\ a\rangle \mid q[\underline{out\ n}.\text{lock}\ p]]] \mid b[Q]$

$\rightarrow a[!(x).l[in\ m.x.n[\langle x\rangle \mid out\ m.q[out\ n.\text{lock}\ p]]] \mid !open\ n.p[\text{count}\ p\ i.P$
$\mid \text{permit lock}] \mid \underline{!open\ q} \mid p[\text{count}\ p\ i'.P \mid \text{permit lock}]$
$\mid m[!open\ l] \mid n[\langle out\ a.in\ b.out\ b.in\ a\rangle] \mid \underline{q}[\text{lock}\ p]] \mid b[Q]$

$\rightarrow a[!(x).l[in\ m.x.n[\langle x\rangle \mid out\ m.q[out\ n.\text{lock}\ p]]] \mid !open\ n.p[\text{count}\ p\ i.P$
$\mid \text{permit lock}] \mid !open\ q \mid \underline{p}[\text{count}\ p\ i'.P \mid \text{permit lock}]$
$\mid m[!open\ l] \mid n[\langle out\ a.in\ b.out\ b.in\ a\rangle] \mid \underline{\text{lock}\ p}] \mid b[Q]$

$\rightarrow a[!(x).l[in\ m.x.n[\langle x\rangle \mid out\ m.q[out\ n.\text{lock}\ p]]] \mid !open\ n.p[\text{count}\ p\ i.P$

$$\mid \text{permit lock}] \mid !open\ q \mid m[!open\ l] \mid n[\langle out\ a.in\ b.out\ b.in\ a\rangle]] \mid b[Q]$$

もし、count $p\ i'.P$ が途中で $i' = 0$ までカウントされていたら、

$$p[\text{count}\ p\ i'.P \mid \text{permit lock}] \rightarrow p[P \mid \text{permit lock}]$$

となり、プロセス P が作動することとなる。

第 7 章

結論

本研究では、意味論の重要性、移動エージェントの概念、Ambinet Calculus の解説にはじまり、主に通信の切断があった場合の移動エージェントの動作の意味を定義してきた。そして、定義したものをリダクションを行なって動作の確認を行なった。

以下に、考察及び今後の課題を述べていく。

7.1 考察

第 5 章、及び第 6 章によって得られた成果は、次の通りである。

- プリミティブな要素のみを持つ Ambient Calculus によって、実際の移動エージェントの動作を表現することができた。
- 上記により、分散計算の振舞いを数学的に調べるためのベースを作ることができた。
- Ambient Calculus 自身の表現力を確認することができた。

7.2 今後の課題

今後の課題として、以下の点が挙げられる。

- 本研究では通信路の切断時における、移動エージェントの動作の形式的モデル・仕様を与えていったが、さらに研究を進めていき、移動エージェントの諸動作の形式的モデル・仕様を与えていかなければならない。

- Mobile Ambients[2] はまだ新しい概念であるため、拡張を行なうなどして機能の強化などを考えていく。

謝辞

本研究を進めるに際し、終始御指導くださった渡部卓雄先生、ならびに折りに触れて有益な助言を与えてくださいました二木厚吉先生に深く感謝致します。そして、研究に関する議論につきあっていただいた言語設計学講座の諸氏に感謝致します。最後に、両親に感謝致します。

参考文献

[1] L. Cardelli. Abstructions for Mobile Computation. Microsoft Research Technical Report MSR-TR-98-34, 1998.

[2] L. Cardelli, A.D. Gordon. Moblile Ambients. in Foundations of Software Science and Computational Structures, Maurice Nivat (Ed.), Lecture Notes in Computer Science, Vol. 1378, Springer, 140-155, 1998.

[3] R. Milner. A Calculus of Communicating Systems. LNCS 92, Spring-Verlag, 1980.

[4] R. Milner, J. Parrow and D. Walker. A culculus of mobile processes, Parts 1-2. Information and Computation, 100(1), 1-77, 1992.

[5] T. Nishigaya. Design of Multi-Agent Programming Libraries for Java. http://blacky.fujitsu.co.jp/hypertext/free/kafka/paper/, 1997.

[6] Tokyo Research Laboratory, IBM Japan, IBM Aglets Software Development Kit White Paper. http://www.trl.ibm.co.jp/aglets/whitepaper/whitepaper-j.html, 1998.