Representation and Analysis of Mobile Computing using
TeleLog—A Mobile Logic Language

SATO, Hiroyuki
Information Technology Center,
The University of Tokyo,
Tokyo, 113-8658, Japan.

ABSTRACT
In the era of the Internet, mobile agents provide novel and promising approach to analysis of computation in the communication network. In this paper, we define TeleLog, and analyze mobile computing using modal logic. go and here primitives are augmented with ordinary logic languages. They are given operational semantics in terms of mobile computing. Moreover, modality of TeleLog is analyzed, and applied to the expressive power analysis of TeleLog. We represent a security mechanism using hook and assert functions that can be represented for the first time by using modality.

Keywords: mobile agent, logic programming, modality, security

1 Introduction
In the era of the Internet, mobile agents provide novel and promising approach to analysis of computation in the communication network. A mobile agent is a computer program that can move around different hosts across the network while carrying code and data. Mobility is observed commonly in internet computing such as Java applets, Web crawling, e-commerce, and some vicious computer virus. Therefore, it is significant to analyze key aspects of mobile computing such as security on a firm semantic basis.

TeleLog[16] is a logic programming approach to mobile agents. Like other logic programming languages, TeleLog is an executable fragment of some kind of logic. There is given a natural logic-based semantic framework for logic languages. In this meaning, behaviors of logic languages are relatively easy to analyze. Our goal is to analyze mobile computing using TeleLog.

This paper presents an approach to the analysis of mobile computing using modal logic. Modal logic is one of major methods in extending logic. Specifically, we give operational semantics to mobile computing in Kripke-like frame. A place corresponds to a world in Kripke’s sense. Mobility means that a program can explicitly move around Kripke worlds. Actually, modality can represent a variety of concepts including time, provability, and belief. This paper tries adding mobility to this menu.

We also show that modality is essential to represent some useful concepts of mobile computation. In particular, security checking mechanism is given a representation in TeleLog. Because security is one of key aspects in mobile computing, we can conclude that this fact implies the usefulness of TeleLog.

The rest of this paper is organized as: Section 2 defines TeleLog, and gives its basic semantics without modality. Section 3 is devoted to the analysis of modality of TeleLog. This analysis is applied to the expressive power analysis of TeleLog in Section 4. Section 5 surveys related work. Section 6 gives a summary of this paper.

2 TeleLog

2.1 Syntax
TeleLog[16] is a logic programming language for mobile calculi. Computation is processed on places. A program can interact with its embracing “places” using go and here primitives.

2.2 Syntax
Definition 1 The syntax of TeleLog is defined in Figure 1. A program is a located Horn clause.

In addition to ordinary Prolog syntax, we consider addresses. go and here are defined as primitives related to addresses.

2.3 Semantics of TeleLog without Modality
First, we give operational semantics to TeleLog without modality. It is given based on resolution on formulas and unification on terms in the same way as ordinary Prolog except go and here. Here, we concentrate on axioms of go and here.
\[ P ::= \text{predicates} \\
M ::= \Diamond P \Box M \\
U ::= \text{unit clause} \\
L ::= \text{literals} \]

\[
U \\
goa \quad \text{go} \\
herea \quad \text{here} \\
C ::= \text{Horn clauses} \\
L ::= \text{goal clause} \\
\text{addr}[C] \\
\]

Figure 1: Syntax of TeleLog

**Definition 2** The axiom of `go` is given as:

\[
l[\langle ? - \text{go} \ m, G_2, \ldots, G_n \rangle \theta] \\
m[\langle ? - G_2, \ldots, G_n \rangle \theta] \\
\]

Here \( \theta \) is a substitution that represents correspondence between variables and terms. Note that in the above definition, a substitution can move from \( l \) to \( m \). This corresponds to collecting terms from multiple addresses.

**Definition 3** The axiom of `here` is given as:

\[
l[\text{here} \ \underline{l}] \\
\]

Because the semantic domain is extended to include addresses, we need extension to resolution and unification.

Because TeleLog is extended so that a program may move among addresses, in a unifier, we may have terms defined on different addresses. Therefore, we extend the unification and the most general unifier (\( \text{mgu} \)) in order to include remote terms. We also denote this extended unifier by \( \text{mgu} \). The mechanism of collecting terms from multiple addresses is a major function of mobile computing.

**Definition 4**

\[
l[\langle ? = (G_1, \ldots, G_n) \phi \rangle] \\
l[\langle ? = (B_1, \ldots, B_m, G_2, \ldots, G_n) \phi \rangle] \\
\]

where \( \theta = \text{mgu}(G_1 \phi, H) \).

\( l \) is the most general unifier of \( G_1 \phi \) and \( H \) at location \( l \).

In this definition, substitutions \( \theta \) and \( \phi \) represent correspondence between variables and terms. \( \text{mgu} \) is calculated on a specific address \( l \), but substitutions can collect terms from multiple addresses as discussed in the definition of `go`.

Actually, the mechanism of `go` and `here` is so general that we can encode a remote subroutine call in which we visit a remote place, do something, and return to the original place.

**Example 1** Consider the program fragment below:

\[
\text{here} \quad \text{Addr, go m, G, go Addr,} \ldots \quad (*) \\
\]

This sequence is an implementation of the remote resolution call provided that the call always succeeds. On any address, this program first moves to \( m \), does some work \( G \), and finally returns to the original address as shown in Figure 2.

### 3 Modality in TeleLog

In this section, we discuss modality in TeleLog.

The resolution scheme for modal logic is already given in [8], and the one for hybrid logic is given in [1]. Even before these schemes, a number of logic programming languages with modality are proposed [11]. This paper adds another resolution scheme for TeleLog, an executable logic with restricted modality. Our scheme can express some important concepts of mobility such as security.

#### 3.1 Operational Semantics

Considering the semantics of TeleLog, the modality must be taken into consideration.

The engine of fully-fledged TeleLog is based on that of TeleLog without modality discussed in Section 2. To discuss modality, we need an extra environment.

**Definition 5** The *modality environment* of TeleLog consists of a pair \( \Theta, \Xi \) of set of clauses. The form

\[
\Theta, \Xi \ni l[C] \\
\]

represents a resolution of TeleLog which satisfies relations in Figure 3.

Intuitively, in the form \( \Theta, \Xi \ni l[C] \), \( \Theta \) represents formulas that wait to be proved at every address where `go` goes,
\[
\frac{\text{here } \text{Addr}, \text{go } m, \text{go } \text{Addr}, \ldots}{l[\text{here } l]}
\]
\[
\frac{\text{go } m, G, \text{go } \text{Addr}, \ldots}{l[\text{go } m, G, \text{go } \text{Addr}, \ldots]}\]
\[
\frac{\text{go } m, G}{m[\text{go } m, G]}\]
\[
\vdots
\]
\[
\frac{m[? - G, \text{go } \text{Addr}, \ldots]}{\sqrt{m[? - G, \text{go } \text{Addr}, \ldots]}\theta[\text{Addr} / l]}\]
\[
\frac{l[? - \ldots]}{\sqrt{l[? - \ldots]}\theta[\text{Addr} / l]}
\]

Figure 2: Resolution associated with remote resolution call: the effect of \( G \) is reflected in the change from \( \theta \) to \( \bar{\theta} \).

1. 
\[
\frac{A \in \Xi}{\Theta, \Xi \vdash l[A]}
\]

2. 
\[
\frac{\Theta, \Xi \vdash l[? - G_1, \ldots, G_n]}{\Theta, \Xi \vdash l[? - (C_1, \ldots, C_m, G_2, \ldots, G_n)\theta]}\]
\[
\theta = \text{mg}u(G_1, H)
\]

3. (modality) 
\[
\frac{\Theta, \Xi \vdash l[? - \Box A, G_1, \ldots, G_n]}{\Theta \cup \{A\}, \Xi \vdash l[? - G_1, \ldots, G_n]}
\]

4. (go) 
\[
\frac{\Theta, \Xi \vdash l[? - \text{go } m, G_1, \ldots, G_n]}{\Theta, \Xi \vdash l[\text{go } m]}\]
\[
\vdots
\]
\[
\frac{\Theta, \Xi \cup \{A_0, \ldots, A_m\} \vdash m[? - G_1, \ldots, G_n]\theta_0 \cdots \theta_m \theta}{\Theta, \Xi \cup \{A_0, \ldots, A_m\} \vdash m[? - G_1, \ldots, G_n]\theta_0 \cdots \theta_m \theta}
\]

Figure 3: Resolution Rules of TeleLog
4 Expressive Power of Modality of TeleLog

The modality in TeleLog can express some significant concepts of mobile computing such as hook and assert. Review the rules of modality and \texttt{go} in Definition 5.

\[
\begin{align*}
\Theta, \exists \alpha \Gamma[l \vdash \neg \Box A, G_1, \ldots, G_n] & \\
\Theta \cup \{A\}, \exists \alpha \Gamma[l \vdash \neg G_1, \ldots, G_n] & 
\end{align*}
\]

If we encounter a modal formula $\Box A$ during a resolution, $A$ must be resolved in every address thereafter visited. Therefore, $A$ must be saved in $\Theta$-part, and when a \texttt{go} is executed, it is resolved. This process is represented in the resolution of $\Theta$-part in the rule of \texttt{go} (4. in Figure 3) which requires both the success of resolution of $\emptyset, \exists \alpha \vdash m[l \vdash \neg \wedge \Theta]$, and adds extra axioms given as $A_0, \ldots, A_m$ from the outer world.

Therefore, we can say that if the resolution succeeds, the check (resolution of $\Theta$-part) is done, and we can move $l$ to $m$ with additional axioms $A_0, \ldots, A_m$. This implements a hook mechanism, which can be used to describe the security check mechanism in mobile calculus.

One more point to be noted in the rule of \texttt{go} is that if $\Box A$ is asserted as a unit clause, then at every address thereafter visited, $A$ is treated as a unit clause in $\Xi$-part. This can be used as side-effect in a resolution.

4.1 Hook and Security Check

Processing $\Box$-ed clauses is a kind of prologues of function calls. Actually, if we register some actions as a set of $\Box$-ed clauses, we can use them as hooking mechanism at entering an address.
If a program is written without `verify_certificate`, the security is not checked. Only not-safety-requiring actions are possible. In this way, we can describe a security layer.

4.2 Assert

Assertions have side effects in the sense that it changes the semantics of a program. \( \Box A \) acts as an assertion of \( A \) in the addresses a program visits. Although this modality is restricted to unit modality, this extension can partially express functions of `assert` primitive.

Particularly, we show that a restricted form of `assert` can be represented in TeleLog. This enables us to describe security mechanism in pure Telelog.

Example 3 Let an address \( \text{LocM} \) be defined as follows:

\[
\text{LocM}[\text{verify_certificate}(X) :-
\text{cert_check_strict}(X).]
\]
\[
\text{LocM}[]\text{safety_checked_in_m}.
\]

Let an address \( M \) be defined as follows:

\[
\text{M}[\text{verify_certificate}(X) :-
\text{safety_checked_in_m}].
\]
\[
\text{M}[\text{verify_certificate}(X) :-
\text{cert_check}(X), \text{LocM}::\text{true}].
\]
\[
\text{M}[\text{cert_req_actionA}(X) :-
\text{safety_checked_in_m}, \text{actionA}(X)].
\]
\[
\text{M}[\text{cert_not_req_actionB}(X) :- \text{actionB}(X)].
\]

If a program enters address \( M \), its certificate is checked, and goes to \( \text{LocM} \). In \( \text{LocM} \), \( \Box \text{safety_checked_in_m} \) is asserted. This means that at the point of the return from \( \text{LocM} \) (the success of \( \text{LocM}::\text{true} \)), \( \text{safety_checked_in_m} \) is valid. At the return, because `verify_certificate` succeeds by the first clause, there is no infinite travel between \( M \) and \( \text{LocM} \).

In this meaning, `assert(A)` in Prolog can be represented as \( \text{LocM}::A \).

This result indicates that security check can be expressed without `assert` primitive.

Note that TeleLog itself is open to the security policy in the real world. For example, the key predicate `safety_checked_in_m` must be confidential to other addresses. Moreover, the connectivity of \( \text{LocM} \) to \( M \), and its dis-connectivity to addresses other than \( M \) must be separately expressed (Figure 5). However, despite these open problems, we can say that TeleLog is a candidate for expressing security mechanism.

5 Related Work

Extensions of logic programming languages and unification to include code mobility have been tried mainly in AI. In particular, BinProlog/Jinni[18, 17] extends Prolog to facilitate communication between agents via global tuple space. Unification in tuple space is critical in its remote resolution.

Addition of modality to logic programming languages has a long history [11]. In its earlier stages, temporal logic was a major concern [3, 4, 15]. A resolution system for modal logic is given in [8]. For hybrid logic, [1] gives a resolution system. Our resolution gives another system. The difference to previous researches lies in that we give a special treatment to modality that enables the representation of hooking, and security mechanism as its application. Besides, logic programming, Modal-ML [20] implements [12]. In Modal-ML, modality is used as representing a stage of compilation.

There have been proposed a number of programming languages such as Telescript[19], Odyssey, Voyager, and Aglets[9] to represent mobility. Among those, Telescript is the first industrial strength mobile agents programming language designed by White. TeleLog owes much to Telescript in the idea of representing mobility.

Another line that has direct connection with this paper is mobile calculus. It has emerged as a variant of process algebra [2], but soon has become one major line [5, 10, 14]. Among those, Dπ[13] has introduced resources together with locations and \( \text{go} \) as ours. Our system is type-free while Dπ is concerned with types for locations, channels, and transmissions. [6] uses locations as layering services in applications, host, and net.

The expanding Internet world requires appropriate calculus for mobile calculus that expresses Java applets or Web crawling [7]. Mobile calculus is also closely related to mobile shopping, an important branch of e-commerce, which Telescript has as background.
6 Concluding Remarks

In Section 2, we have defined TeleLog, and gives its basic semantics without modality. go and here primitives are introduced to ordinary logic languages. They are given operational semantics in terms of mobile computing. In Section 3, we have focused on modality of TeleLog. Its operational semantics is given. This analysis has been applied to the expressive power analysis of TeleLog in Section 4. We have represented a security mechanism using hook and assert functions that can be represented first by using modality.

References