A MODEL FOR POLICY BASED SERVICE COMMUNITY

Hironobu Kuruma
Systems Development Laboratory, Hitachi, Ltd.
1099 Ohzenji, Asao, Kawasaki, Japan

Shinichi Honiden
National Institute of Informatics/ The University of Tokyo
2-1-2 Hitotsubashi, Chiyoda, Tokyo, Japan

Keywords: access control, policy of community, federation, web services.

Abstract: Since the World Wide Web is an open system, it is difficult to maintain the information about services on the Web in a centralized server. Therefore the service mediation system could be constructed by federation of service communities, in which each community provides and mediates limited number of services according to its own policy. The federation should preserve the policy of each community. Furthermore, (1) scalability, (2) verifiability of policy compliance, and (3) flexibility to the change of federation relation should be considered in implementing the federation. In this paper, we introduce a notion of policy of community based on access control among players and show a community model that is aimed at specifying communications between players compliant with policy. The community model provides function specification of the service mediation system. Since a meta-architecture based language is used to describe community model, communications for the cooperation of communities can be represented separately from the communications for service request and provision. As the result, our community model (1) represents communications between players in a modular way, (2) provides a basis for verification of policy compliance, and (3) encapsulates the dependencies on partner communities.

1 INTRODUCTION

Service provision, usage, and mediation on the web (Toyouchi et al., 2000) are becoming popular in recent years. Since the web is an open system, there may be much difference in the quality and trustworthiness of the services. Therefore, the users have to carefully read the agreements on usage of the service and accept them before using these services. But it is not easy to understand the agreement for every service usage, and ordinary users may skip reading and take risks of privacy loss etc., or abandon the service usage.

A solution of this problem is to construct a community which is a gathering of players such as service providers, users, and mediators under a policy. In such a community, every member provides, uses, and mediates services without agreements for each service usage as far as their behaviours are validated by the policy. For example, let us consider a hotel reservation system shown in Figure1. In this system, a customer requests the mediator to reserve a room, and the mediator reserves a room in a suitable hotel on behalf of the customer. If the system handles the messages only from the customers, mediator, and hotels that agreed a certain policy, a community is formed in which players trust the partners in the scope of the policy. The communities may federate to construct a larger community. Assume a flight reservation system shown in Figure2. In this system, a customer asks the recommender the flight which fits his requirements. And then the customer request the airline company recommended by the recommender to reserve the flight. In the federation of the hotel reservation community and the flight recommendation community, how the system is constructed so that the customers reserve rooms as well as flights? The federation should preserve the policy of each community. Furthermore, (1) scalability, (2) verifiability of policy compliance, and (3) flexibility to the change of federation relation are considered in implementing it.

In this paper, we describe a model for constructing a policy based community. Though Role-Based Access Control (Ferraiolo et al., 2001; Ferraiolo et al., 2003) is a powerful approach for providing security in a community, we focus on the secutiry among communities. Our objective is to (1) introduce the notions of policy of community and policy of federation in-

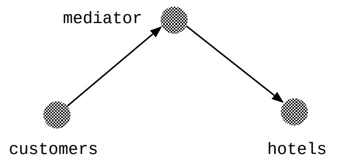

Figure 1: Hotel reservation service

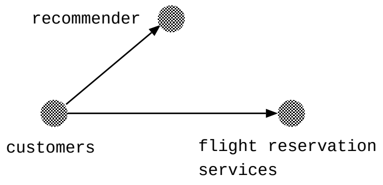

Figure 2: Flight reservation servce

dependent of the policy of each player, (2) show a model for implementation of federation, and (3) provide a basis for construction and verification of policy based service provision/usage/mediation systems. The notion of the policy of community is introduced in section 2 as an accessibility graph of player's roles. Based on the accessibility graph, policy preserving federation is also defined. To support the implementation of policy based federation, we introduce a community model in section 3 and show an example in section 4. We show related work in section 5 and conclude our approach in section 6.

2 ACCESS POLICY

To describe accessibility of players in a community independent of individual players, we introduce a notion of the policy of community.

2.1 Agent

In this paper, we refer to entities participating in the service provision, usage, and mediation on the web as agents. An agent may be a computer system which processes services automatically, or a person who handles the requests of customers via web. In the following, Agn denotes the set of all agents.

2.2 Role

A role is a name of task that agents have for provision, usage, or mediation of services. Typical roles are service provider, requester, and mediator. Let Rle denote the set of all roles and

$$g : Agn \rightarrow 2^{Rle}$$

be a function which relates each agent to the set of roles it plays.

2.3 Community

A community is a set of agents playing a role in provision, usage, and mediation of services. An agent can be a member of more than one community at the same time. In the following, Cmm denotes the set of all names of communities, and R_c represents the set of all roles of community which has name $c \in Cmm$.

Let A_c be a community of name $c \in Cmm$:

$$A_c = \{x | g(x) \cap R_c \neq \emptyset\}.$$

We assume that an agent plays at most one role in a community at a time. That is;
for all $a \in Agn$ and $c \in Cmm$, if there exist X, Y such that $X, Y \in R_c \wedge X, Y \in g(a)$, then $X = Y$.

2.4 Policy of Agent

We define the policy of each agent as binary relation of agents $p \subseteq Agn \times Agn$. For agents $x, y \in Agn$, $(x, y) \in p$ represents that x is accessible to y.

2.5 Policy of Community

For each community $c \in Cmm$, we define the policy of community c as binary relation $p_c \subseteq R_c \times R_c$. The policy of community p_c represents the accessibility relationship of roles. That is;
for all $X, Y \in R_c$, X is accessible to Y if and only if $(X, Y) \in p_c$.

Let p be the policy of agents and p_c be the policy of community c. For all $x, y \in Agn$, x is accessible to y if and only if there exist X and Y such that $(x, y) \in p \wedge X \in g(x) \wedge Y \in g(y) \wedge (X, Y) \in p_c$.

2.6 Policy of Federation

A federation is a community of which roles are superset of roles of communities participating in the federation. In this paper, we define policy of federation as the delegation relation of access rights between roles. The policy of federation provides the basis for verifying the preservation of communities' policies participating in the federation.

For communities m and n participating in a federation, we define the policy of federation as a pair $P_{mn} \subseteq R_m \times R_n$. $(X, Y) \in P_{mn}$ represents that the rights of X to access the roles in m are delegated to Y. That is, Y is accessible to the roles of m to which X is accessible.

To extend the notion of policy of federation to federation of more than two communities, we introduce

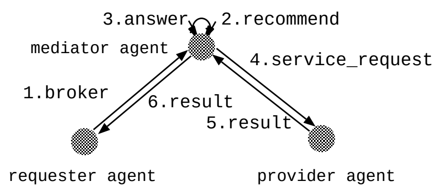

Figure 3: Communication for brokerage

reflective transitive closure P^*. For all pairs of component communities m and n of a federation, P^* is the minimal set satisfying the following.

- $P_{mn} \subseteq P^*$ and $P_{nm} \subseteq P^*$
- $\forall X \in R_m \cup R_n$, $(X, X) \in P^*$
- for all roles X, Y, Z, if $(X, Y) \in P^* \wedge (Y, Z) \in P^*$ then $(X, Z) \in P^*$

2.6.1 Policy Preservation

Assume that $c_1, ..., c_m$ are communities participating in federation c. Let $p_1, ..., p_m$ be the policy of each component community and p_c be the policy of community c. When p_c satisfies the following, we say that the policy of c preserves the policy of $c_i (1 \leq i \leq m)$ under the policy of federation P^*.

- $p_i \subseteq p_c$
- for role Y of c_i, if $(X, Y) \in p_c$ then there exists X' such that $(X', X) \in P^* \wedge (X', Y) \in p_i$

2.7 Example

2.7.1 Policy of Community

Let A be a community of brokerage in which agents interact as shown in Figure 3; that is

1. a requester agent requests a service to the mediator agent
2. the mediator agent requests the service to a provider agent
3. the mediator agent provides the requester agent for the service on behalf of the provider agent

On the other hand, let B be a community of recommendation in which agents interact as shown in Fig 4; that is

1. a requester agent requests a service to the mediator agent
2. the mediator agent recommends a provider agent to the requester agent
3. the requester agent requests the service to the provider agent

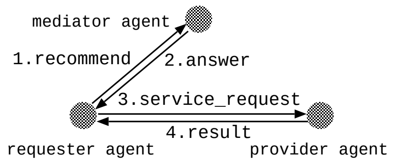

Figure 4: Communication for recommendation

Let $A.R$, $A.M$, and $A.P$ be the roles of requester agents, mediator angents, and provider agents of the community A respectively. The policy of community A is

$$p_A = \{(A.R, A.M), (A.M, A.P), (A.M, A.M)\}$$

Similarly, the policy of community B can be represented as follows, where the roles $B.R$, $B.M$ and $B.P$ represent the role of requester, mediator and provider agents respectively.

$$p_B = \{(B.R, B.M), (B.R, B.P)\}$$

2.7.2 Policy of Federation

In the federation of A and B, let us require that $B.R$ has access right that $A.R$ has to the roles of A and $A.M$ has access right that $B.R$ has. The policy of federation P_{AB}, P_{BA} and P^* are as follows.

$$P_{AB} = \{(A.R, B.R)\}, P_{BA} = \{(B.R, A.M)\}.$$

$$\begin{aligned} P^* = \; & P_{AB} \cup P_{BA} \cup \\ & \{(A.R, A.R), (A.M, A.M), (A.P, A.P), \\ & (B.R, B.R), (B.M, B.M), (B.P, B.P)\} \end{aligned}$$

2.7.3 Policy Preservation

Let D be the community constructed by the federation of A and B, and take the policy of the community D

$$\begin{aligned} p_D = \; & p_A \cup p_B \cup \\ & \{(A.M, B.M), (A.M, B.P), (B.R, A.M)\} \end{aligned}$$

Figure 5 shows the policy of D. In this figure, the circles represent roles and the dotted arrows represent the policy of federation. The solid arrows represent the policy of D. For example, an arrow from X to Y means that $(X, Y) \in p_D$. With respect to the definition in section 2.6.1, we can see that the policy of A and B are preserved in p_D.

2.7.4 Extension of Federation

The federation may be extended by the further federation with other communities. Let C be a community of recommendation with the policy of community p_C.

$$p_C = \{(C.R, C.M), (C.R, C.P)\}$$

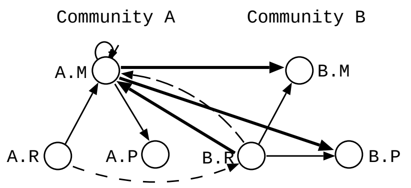

Figure 5: Policy preservation

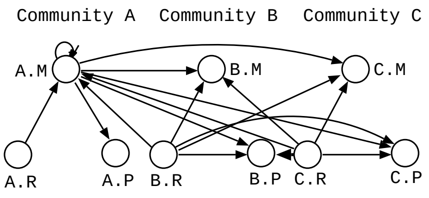

Figure 6: Policy of community extended by further federation

Assume the policy of federation;

$$\begin{aligned} P_{CF} &= \{(C.R, B.R), (C.R, A.M)\}, \\ P_{FC} &= \{(A.R, C.R), (B.R, C.R)\}. \end{aligned}$$

The following policy p_E of community E, which is the federation of C and D, is a policy which preservies the policies of C and D. p_E also preserves the policies of A and B.

$$\begin{aligned} p_E = \; & p_D \cup p_C \cup \{(C.R, A.M), \\ & (C.R, B.M), (C.R, B.P), (A.M, C.M), \\ & (A.M, C.P), (B.R, C.M), (B.R, C.P)\} \end{aligned}$$

Figure 6 shows the policy of community E.

3 COMMUNITY MODEL

In the federation, the agents of each community are required to be compliant with the policy of the community when they interact with agents of other communities. Therefore, the implementation of a policy based federation basically consists of restricting the communications across the communities and adding communications for cooperation of communities.

To support the implemention of the federation, we introduce a community model which represents the communications in federation hierachically. This model is written in a modelling language whose syntax is shown in Appendix A. The modelling language introduces the notion of meta-hierarchy and field; (1) meta-level description can be attached to specify the dynamism of agent's behaviour, and (2) message passing is restricted to the range of agents which share the same field.

3.1 Meta-level Architecture

In the modelling language, an agent is specified in several levels. The lowest level is 0 level, which we call the base level. Though the modelling language does not fix the number of meta-levels, let n denotes the highest level declared in the community model. In the syntax of Appendix A, the number of "m"s at `<DESCRIPTION ID>` show the meta-level. The behaviour of agent, such as change of state and transmission of a message, is specified in each level.

A message transmitted in level i ($0 \leq i < n$) is manipulated in level $i+1$; i.e. we can control the transmission of the message by explicitly specifying the process of dequeueing, passing to the receiver agent, and enqueing to the message queue in $i+1$ level. The invocation of method in i level is performed by `execute` method of $i+1$ level. Below we describe the meaning of message invocation, where N denotes set of natural numbers representing meta-level, Msg denotes set of message patterns, and Fld denotes set of fields.

Firstly, we define a function

$$acceptable : Msg \times Agn \times N \rightarrow bool$$

which shows the existence of a method.

- if a method which fits in message m is specified in i level of agent a : $acceptable(m, a, i) = true$
- otherwise : $acceptable(m, a, i) = false$

Since any message transmitted in i level can be manipulated in $i+1$ level, the message invocation by message m in $i \leq n$ level is decided by the following function

$$executable : Msg \times Agn \times N \rightarrow bool.$$

- $i = n$: $executable(m, a, i) = acceptable(m, a, i)$
- $i < n$: if the invocation of `execute` for m is specified or nothing is specified (default assumed) in $i+1$ level, $executable(m, a, i) = acceptable(m, a, i)$
  if the message m is rewritten to m' and `execute` is invoked for m' in $i+1$ level, $executable(m, a, i) = acceptable(m', a, i)$
  otherwise : $executable(m, a, i) = false$

3.2 Message Transfer

A community model possibly contain several fields. The message passing of agents are limited by the fields; i.e. no message is passed from an agent to other agents unless they stay in the same field. However, a message can be transferred across the fields if it is forwarded by agents placed in more than two fields.

Consider that an agent a_1 in field f_1 sends an i-level message m to a_2 in field f_2. Due to the message manipulation specified in $i+1$ level, this message may be passed to other agents and executed by them. We define the function $transfer$:

$$Msg \times (Agn \times Fld) \times (Agn \times Fld) \times N \rightarrow bool$$

which shows the transformation of the message to an agent a in f as follows.

$$transfer(m, (a_1, f_1), (a, f), i) =$$
$$forward(wrap(m), (a_1, f_1), (a, f), i + 1) \wedge$$
$$executable(m, a, i)$$

where $wrap(m_i)$ denotes the message which passes the message m_i in $i+1$ level. If $wrap(m_i)$ is not specified in $i+1$ level, default method is assumed so as to $acceptable(wrap(m_i), a, i + 1) = true$.

The function $forward$:

$$Msg \times (Agn \times Fld) \times (Agn \times Fld) \times N \rightarrow bool$$

which shows the forwarding of message m transmitted from a_1 in f_1 to a_2 in f_2 in i level, is defined as follows.

- $i = n + 1$
  if $f = f_1 = f_2$ and $a = a_2$ then
  $forward(m, (a_1, f_1), (a, f), i) = true$
  else $forward(m, (a_1, f_1), (a, f), i) = false$
- $0 < i \leq n$
  if there exists an agent a_3 which forwards m,
  $$forward(m, (a_1, f_1), (a, f), i) =$$
  $$forward(m, (a_1, f_1), (a_3, f_3), i) \wedge$$
  $$transfer(m, (a_3, f_4), (a, f), i)$$
  otherwise
  $$forward(m, (a_1, f_1), (a, f), i) =$$
  $$transfer(m, (a_1, f_1), (a, f), i)$$

where f_3 and f_4 are fields in which a_3 stays, and they may be different.

3.3 Verification of Compliance with Policy

The communications specified in the community model should be compliant with the policy of the community. This can be verifyed as follows.

Let us define a relation Acc which shows the access between roles.

$$Acc = \{\langle X, Y\rangle | \exists a, b \in Agn(\exists u, v \in Fld$$
$$(\exists m \in Msg(X \in g(a) \cap R_U \wedge Y \in g(b) \cap R_V$$
$$\wedge transfer(m, (a, u), (b, v), 0))))\}$$

where R_U and R_V are the set of roles of communities U and V respectively.

For the policy of community p_c, the community model is an implementation of p_c if and only if

$$Acc \subseteq p_c.$$

4 EXAMPLE OF COMMUNITY MODEL

In this section, we show a community model for the community illustrated in section 2.7.

4.1 Model of Each Community

In the community model, let A and B be fields and place the typical agents which represent roles of communities A and B in fields A and B respectively. The communications shown in Figures 3 and 4 are performed in the base-level of each agent. On the other hand, we described the communications for cooperation of communities in the meta-level of each agent as follows.

The following are communications for cooperation that are common in every mediator agents specified in the 1 level.

- forward the message to the receiver agent which shares the same field
- forward the message to the mediator agent if the receiver agent is not in the same field

For the requester and provider agents in community B, the following is specified in the 1 level to communicate agents in other community directly.

- ask the mediator agent for forwarding the message if the receiver agent is not in the same field

The following are specified in the 1 level of mediator agents in community A which behave as provider agents when they receive request from agent in other communities.

- manipulate the base-level message `answer` so as to reply itself as a provider
- change the base-level message `service_request` to `broker` message

4.2 Model of Federation

The community D, which is the federation of A and B, is modelled as follows.

- add mediator agents of community A to field B
- add mediator agents of community B to field A
- to delegate a request to other mediator, a mediator which received a `recommend` message in base level performs
  - execute the message at base level
  - if the request is satisfied, return `answer` message
  - otherwise forward the `recommend` message to other mediators

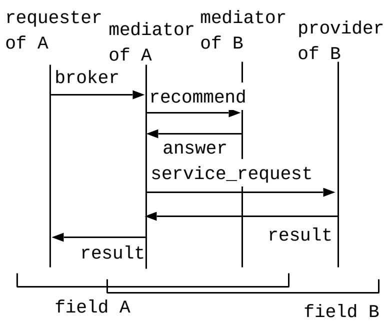

Figure 7: Message sequence (A to B)

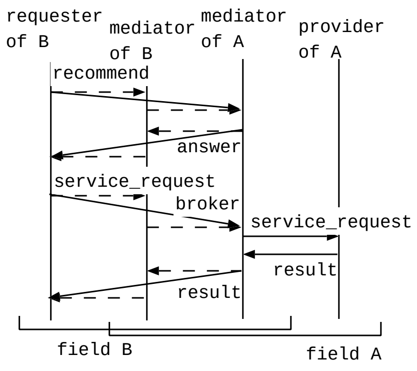

Figure 8: Message sequence (B to A)

- change the common communication in the 1 level of the mediator agents of A so that they reject the message transfer from agents in other field, if the receiver of the message is requester or provider agent in field A.

Figure 7 shows the communication of a requester agent of A and agents in B when the requester agent uses a service in B. Figure 8 shows the communication of requester agent of B and agents in A when the requester agent uses a service in A. In this figure, dotted arrows show the communication in the 1-level.

4.3 Extension of Federation

The community E, which is a federation of communities D and C, is modelled as follows.

- add mediator agents of community A to field C
- add mediator agents of community C to field A
- add the following behaviours to agents of C when they receive a `recommend` message
  - execute the message at base level.
  - if the request is satisfied, return `answer` message.

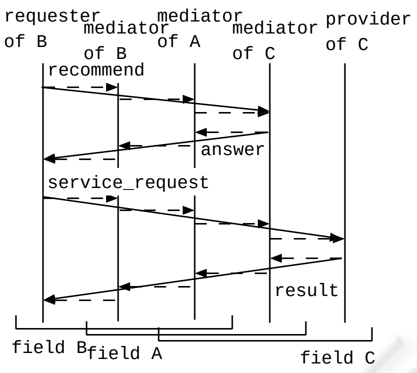

Figure 9: Message sequence (B to C)

  - otherwise forward the `recommend` message to other mediators.

Figure 9 shows the communication of requester agent of B and agents in C when the requester agent uses a service in C.

5 RELATED WORK

RBAC/Web (Kuhn et al., 1997) is an implementation of Role Based Access Control for Web services. It controls the access to the information on the Web based on the roles assigned to the users. The concept that operations are associated with roles simplifies the understanding and management of privileges. However, RBAC/Web does not support the federation we described in this paper.

UDDI, Universal Description, Discovery, and Integration, is a registry for sharing information on Web Services. It is also specification for service description and discovery. In the version 3 of the specification, UDDI support the multiple registry interaction and policy description for different registry implementations. In our work, the policy of community defines the accessibility relations in service provision/usage/mediation community and the community model specifies the communications between players. Therefore the UDDI registry can be used a basis for the implementation of communities and their federations in our work.

Many works have been done in the research domain of meta-level programming and reflection (Maes, 1988; Charlton, 1996). Adaptive Software Agents behave reflectively to adapt the change of their environments. The objective of these works is principally to program agents that adopt their environments. The objective of our work, on the other hand, is to provide a method for developing service mediation systems that are flexible with the change of federation relations. The agents in our community model provide function specification of a service mediation system which is to be implemented in ordinary programming language. In the future, agents in a community model may be implemented on the basis of adaptive software agents so that they autonomously adapt and observe the policies.

6 CONCLUSION

In this paper, we introduced a notion of policy of service community based on access control among roles and showed a multi-level communication model for policy implementation. Our model (1) represents communications in a hierarchcal way, (2) provides a basis for verification of policy preserving communications and (3) encapsulates the dependencies on partner communities.

Since the agents of our model are described in multi-level, implementing them in ordinary programming language is not an easy task. Our future work is to develop a method for constructing service mediation systems from our model.

REFERENCES

Charlton, P. (1996). Self-configurable software agents. In *Advances in Object-Oriented Metalevel Architectures and Reflection*, pages 103–127. CRC Press.

Ferraiolo, D. F., Kuhn, D. R., and Chandramouli, R. (2003). *Role-Based Access Control*. ARTECH HOUSE, Inc.

Ferraiolo, D. F., Sandhu, R., Gavrila, S., Kuhn, D. R., and Chandramouli, R. (2001). Proposed nist standartd for role-based access control. *ACM Trans. on Information and System Security*, 4(3):224–274.

Kuhn, D., Barkley, J., Cincotta, V., Ferraiolo, D., and Gavriella, S. (1997). Role based access control for the world wide web. In *Proc. 20th National Information Systems Security Conference*, pages 331–340.

Maes, P. (1988). Issues in computational reflection. In *Meta-Level Architectures and Reflection*, pages 1–35. Elsevier Science Publishers B.V. (North-Holland).

Toyouchi, J., Funabashi, M., and Strick, L. (2000). Service integration platform based on tina 3-tier model and interfaces. In *TINA 2000 CONFERENCE Conference Proceedings*, pages 21–26.

A SYNTAX OF MODELLING LANGUAGE

```
<AGENT> ::=
  <DESCRIPTION> ("," <DESCRIPTION>)*
<DESCRIPTION> ::=
  "{" <DESCTIPTION ID> "|"
  [ "attribute" <ATTRIBUTE>
    ("," <ATTRIBUTE>)* ";;" ]
  [ "method" <METHOD>
    ("," <METHOD>)* ";;" ] "}"
<DESCRIPTION ID> ::=
  agent_name
  | "m(" <DESCRIPTION ID> ")"
<ATTRIBUTE> :=
  variable_name ":" <TERM>
<METHOD> ::=
  <MESSAGE PATTERN> ":" <STATEMENTS>
<MESSAGE PATTERN> ::=
  message_name <TERM>*
<STATEMENTES> ::=
  <STATEMENT> (";" <STATEMENT>)*
<STATEMENT> ::=
  <CONDITIONAL> | <CASE>
  | <ASSIGNMENT> | <EXPRESSION>
<CONDITIONAL> ::=
  "if" <EXPRESSION>
  "then" "(" <STATEMENTS> ")"
  [ "else" "(" <STATEMENTS> ")" ]
<CASE> ::=
  "case" <EXPRESSION> "of"
  (<PATTERN> ":" "(" <STATEMENTS> ")")+
  ["otherwise"":""("<STATEMENTS>")"]
<ASSIGNMENT> ::=
  "let" <PATTERN>
<EXPRESSION> ::=
  <MESSAGE TRANSMISSION> | <TERM>
  | <PATTERN EXPRESSION>
  | "forall" variable_name
    "with (" <EXPRESSION> ")"
    <EXPRESSION>
  | "forsome" variable_name
    "with (" <EXPRESSION> ")"
    <EXPRESSION>
<MESSAGE TRANSMISSION> ::=
  <TERM> "<-" <MESSAGE PATTERN>
<TERM> ::=
  "(" <EXPRESSION> ")"
  | agent_name "." field_name
  | variable_name | "dequeue"
  | "execute(" variable_name ")"
<PATTERN EXPRESSION> ::=
  <PATTERN> "=" <PATTERN>
<PATTERN> ::=
  <EXPRESSION> | literal
```