A Logic Specification for Usage Control

Xinwen Zhang, Jaehong Park
Francesco Parisi-Presicce, Ravi Sandhu

George Mason University

SACMAT 2004

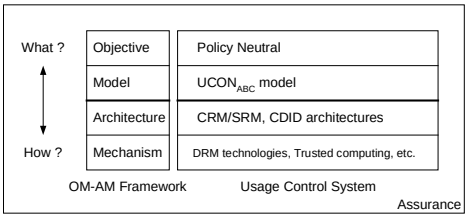
UCON

- UCON provides a general model beyond DRM and Trust management:
  - Digital Rights Management (DRM)
    - Mainly focus on intellectual property rights protection with architecture and mechanism level studies
    - Lack of access control model
  - Trust Management
    - Authorization for strangers' access based on credentials
    - Lack of an abstract model with attribute-based.

Outline

- Introduction of UCON
- Temporal Logic of Action (TLA)
- Logic Model for UCON with TLA
- Specification of Authorization Core Models
- Obligation and Conditions
- Conclusions and Future Work

OM-AM Layered Approach



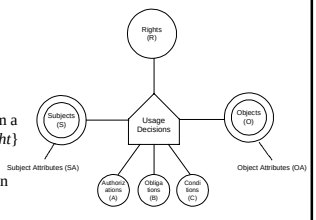
- Model examples: Access Matrix, Lattice-based model, Role-base access control model

UCON

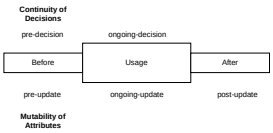
- A unified framework for next generation access control
- A comprehensive model to represent the underlying mechanism of existing access control models and policies.
- Try to extend the limits of traditional access control models:
  - Authorization only – No obligation or condition based control
  - Identity based only – No attributes based support
  - Decision is made before access – No ongoing control
  - No consumable rights - No mutable attributes
  - Rights are pre-defined and granted to subjects

UCON Model

- Basic components:
  - Subjects and attributes
  - Objects and attributes
  - Rights
- Logically, UCON is a mapping from a set of {*subject/object attributes, right*} to {*true, false*}
- Usage control decisions are based on authorization, obligations, and conditions.
- Referred as *UCON$_{ABC}$* model



Continuity and Mutability of UCON



- A single usage process has three phases
  - before access, during usage, and access
- Continuity: control decision can be checked before or during access
- Mutability: attribute updates can be performed before, during or after access
  - Pre-update, on-update, and post-update

Temporal Logic of Action

- Basic Terms:
  - Variables: *x, y*
  - Values: 5, "abc"
  - Constants
  - A state is an assignment of values to variables
- Functions: nonboolean expression with variables and constants
  - Semantically, a function is a mapping from states to values.
- State Predicates: boolean expression with variables and constants
  - Semantically, a predicate is a mapping from states to booleans.
- Actions: boolean expression with variables, primed variables, and constants
  - Semantically, an action is a function assigning a boolean to a pair of states *(s,t)*, where *s* is the old state with variables, and *t* is the new state with primed variables.

Core Authorization Models

- According to the authorization control attribute update points, we have seven core authorization models:
  - ***preA*$_0$**: control decision is determined before access, and there is no attribute update.
  - ***preA*$_1$**: control decision and and attribute update before access.
  - ***preA*$_3$**: control decision is determined before access, and attribute update after access.
  - ***onA*$_0$**: control decision is checked and determined during usage, and there is no attribute update.
  - ***onA*$_1$**: control decision is checked and determined during usage, and there is attribute update before access.
  - ***onA*$_2$**: control decision is checked and determined during usage, and there is attribute update during usage.
  - ***onA*$_3$**: control decision is checked and determined during usage, and there is attribute update after usage.
- A real UCON system may be a hybrid of them.

TLA

- Behavior: a sequence of states

  <s0, s1, s2,…,>

- Semantics of an action *A*:

$$< s_0, s_1, s_2, ... > [\![A]\!] \equiv s_0 [\![A]\!] s_1$$

- Temporal operator: □(always)

$$< s_0, s_1, s_2, ... > [\![\Box A]\!] \equiv \forall n \in N : s_n [\![A]\!] s_{n+1}$$

- Temporal Formula:

$$F :\equiv < predicate > | \Box < action > | \neg F | F \wedge F | F \vee F | F \rightarrow F | \Box F$$

- Semantics:

$$< s_0, s_1, s_2, ... > [\![F]\!] \equiv s_0 [\![F]\!] s_1$$
$$< s_0, s_1, s_2, ... > [\![\Box F]\!] \equiv \forall n \in N :< s_n, s_{n+1}, s_{n+2}, ... > [\![F]\!]$$

Outline

- Introduction of UCON
- Temporal Logic of Action (TLA)
- Logic Model for UCON with TLA
- Specification of Authorization Core Models in UCON
- Obligation and Conditions
- Conclusions and Future Work

TLA

- Other temporal operators:
  - "Eventually":

$$< s_0, s_1, s_2, ... > [\![\Diamond F]\!] \equiv \exists n \in N :< s_n, s_{n+1}, s_{n+2}, ... > [\![F]\!]$$

$$\Diamond F \equiv \neg \Box \neg F$$

  - "Next":

$$< s_0, s_1, s_2, ... > [\![\bigcirc F]\!] \equiv s_1 [\![F]\!] s_2$$

  - "Until":

$$< s_0, s_1, s_2, ... > [\![FUG]\!] \equiv \exists i \geq 0 : (s_i [\![G]\!] s_{i+1} \wedge (0 \leq j \leq i \rightarrow s_j [\![F]\!] s_{j+1}))$$

- Past temporal operators:
  - *Has-always-been, Once, Previous, Since*

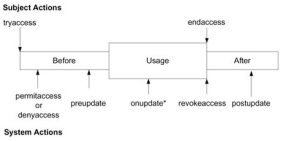
$$< ..., s_{-2}, s_{-1}, s_0, s_1, s_2, ... > [\![\blacksquare F]\!] \equiv \forall n < 0 : s_n [\![F]\!] s_{n+1}$$
$$< ..., s_{-2}, s_{-1}, s_0, s_1, s_2, ... > [\![\blacklozenge F]\!] \equiv \exists n < 0 : s_n [\![F]\!] s_{n+1}$$
$$< ..., s_{-2}, s_{-1}, s_0, s_1, s_2, ... > [\![\ominus F]\!] \equiv s_{-1} [\![F]\!] s_0$$
$$< ..., s_{-2}, s_{-1}, s_0, s_1, s_2, ... > [\![FSG]\!] \equiv \exists i < 0 : (s_i [\![G]\!] s_{i+1} \wedge (i < j < 0 \rightarrow s_j [\![F]\!] s_{j+1}))$$

Outline

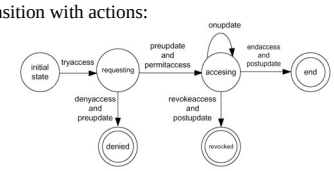
- Introduction of UCON
- Temporal Logic of Action (TLA)
- Logic Model for UCON with TLA
- Specification of Authorization Core Models in UCON
- Obligation and Conditions
- Conclusions and Future Work

Logic Model of UCON: Actions

- Two types of actions:
  - Actions performed by a subject
  - Actions performed by the system



- *state(s,o,r)* transition with actions:



Logical Model of UCON: Attributes

- A state of UCON is an assignment of values to attributes:
  - Subject attributes: role, security clearance, credit amount, etc.
  - Object attributes: type, directory, etc.
  - System attributes: time, location, etc.
  - A special system attribute:
    - *state(s,o,r)={initial, requesting, denied, accessing, revoked, end}*
    - To specify the status of a single access process *(s,o,r)*
    - Authorization actions defined to change this state.

Logic Model of UCON

DEFINITION 1. *A logical formula in UCON is defined by the following grammar in BNF:*

$$\phi ::= a|p(t_1, ..., t_n)|(\neg\phi)|(\phi\wedge\phi)|(\phi \rightarrow \phi)|\forall x : \phi|\exists x : \phi|\Box\phi|\Diamond\phi|\bigcirc\phi|\phi\mathcal{U}\phi|\blacksquare\phi|\blacklozenge\phi|\ominus\phi|\phi\mathcal{S}\phi|$$

where a is an action, p is a predicate of arity n, t_1, ..., t_n are terms, and x is a variable.

DEFINITION 2. *An logic authorization model of UCON is a triple:$\mathcal{M} = (\mathcal{S}, \mathcal{P}, \mathcal{A})$, where*

- *$\mathcal{S}$ is a sequence of states of subject, object, and the system attributes,*
- *$\mathcal{P}$ is a finite set of state predicates on subject and/or object attributes,*
- *$\mathcal{A}$ is a finite set of state actions.*

Logical Model of UCON: Predicates

- Predicates: boolean expression built from subject attributes, object attributes, and system attributes:
  - Unary predicates:
    Alice.credit > $1000, file1.classification = “secure”
  - Binary predicates:
    Dominate(Alice.clearance, file1.classification)
    in((Bob, read), file2.ACL)
  - Ternary predicate *permit(s,o,r)*:
    - usage control decision
    - *True* if a *s* is allowed to access *o* with *r*.

Logical Model of UCON

If a model $\mathcal{M}$ with a state s satisfies a formula ϕ, we write $\mathcal{M}, s \vDash \phi$. Semantically,

1. $\mathcal{M}, s_0 \vDash p$ iff $s_0[p]$, where $p \in P$.
2. $\mathcal{M}, s_0 \vDash a$ iff $s_0[a]s_1$, where $a \in A$, and s_1 is next state of s in S.
3. $\mathcal{M}, s_0 \vDash \neg\alpha$ iff $\mathcal{M}, s_0 \nvDash \phi$.
4. $\mathcal{M}, s_0 \vDash \phi_1 \wedge \phi_2$ iff $\mathcal{M}, s_0 \vDash \phi_1$ and $\mathcal{M}, s_0 \vDash \phi_2$.
5. $\mathcal{M}, s_0 \vDash \phi_1 \rightarrow \phi_2$ iff $\mathcal{M}, s_0 \nvDash \phi_1$ or $\mathcal{M}, s_0 \vDash \phi_2$.
6. $\mathcal{M}, s_0 \vDash \forall x : \phi$ iff for all a, $\mathcal{M}, s_0 \vDash \phi(x/a)$.
7. $\mathcal{M}, s_0 \vDash \exists x : \phi$ iff for some a, $\mathcal{M}, s_0 \vDash \phi(x/a)$.
8. $\mathcal{M}, s_0 \vDash \Box\phi$ iff $\forall n \geq 0 : \mathcal{M}, s_n \vDash \phi$.
9. $\mathcal{M}, s_0 \vDash \Diamond\phi$ iff $\exists n \geq 0 : \mathcal{M}, s_n \vDash \phi$.
10. $\mathcal{M}, s_0 \vDash \bigcirc\phi$ iff $\mathcal{M}, s_1 \vDash \phi$.
11. $\mathcal{M}, s_0 \vDash \phi_1 \mathcal{U} \phi_2$ iff $\exists i \geq 0 : \mathcal{M}, s_i \vDash \phi_2 \wedge (0 \leq j < i \rightarrow \mathcal{M}, s_j \vDash \phi_1)$
12. $\mathcal{M}, s_0 \vDash \blacksquare\phi$ iff $\forall n \leq 0 : \mathcal{M}, s_n \vDash \phi$.
13. $\mathcal{M}, s_0 \vDash \blacklozenge\phi$ iff $\exists n \leq 0 : \mathcal{M}, s_n \vDash \phi$.
14. $\mathcal{M}, s_0 \vDash \ominus\phi$ iff $\mathcal{M}, s_{-1} \vDash \phi$.
15. $\mathcal{M}, s_0 \vDash \phi_1 \mathcal{S} \phi_2$ iff $\exists i \leq 0 : \mathcal{M}, s_i \vDash \phi_2 \wedge (i < j \leq 0 \rightarrow \mathcal{M}, s_j \vDash \phi_1)$

Outline

- Introduction of UCON
- Temporal Logic of Action (TLA)
- Logic Model for UCON with TLA
- Specification of Authorization Core Models
- Obligation and Conditions
- Conclusions and Future Work

Specification of Core Models

- *preA*$_3$:

$p_1 \wedge ... \wedge p_i \rightarrow permit(s,o,r)$
$permitaccess(s,o,r) \rightarrow \blacklozenge(tryaccess(s,o,r)) \wedge permit(s,o,r)$
$endaccess(s,o,r) \rightarrow \Diamond(postupdate(attribute))$

Example 5 DRM membership-based application:
subject: Alice, with attributes of ID and total $expense$
object: book1, with attributes of $title$ and $readingCost$
subject: readingGroup, with attribute $readerList = \{ID1, ID2, ...\}$ and $bookList = \{book1.title, book2.title, ...\}$
right:read

$in(Alice, readingGroup.readerList) \wedge in(book1.title, readingGroup.bookList) \rightarrow permit(Alice, book1, read)$
$permitaccess(Alice, book1, read) \rightarrow \blacklozenge(tryaccess(Alice, book1, read)) \wedge permit(Alice, book1, read)$
$endaccess(Alice, book1, read) \rightarrow \Diamond(postupdate(Alice.expense))$
$postupdate : Alice.expense' = Alice.expense + ebook1.readingCost$

Specification of Core Models

- *preA*$_0$:

$p_1 \wedge ... \wedge p_i \rightarrow permit(s,o,r)$
$tryaccess(s,o,r) \wedge permit(s,o,r) \rightarrow \bigcirc(permitaccess(s,o,r))$

- Example 2: BLP model

$dominate(s.cleareance, o.classfication) \rightarrow permit(s,o,read)$
$tryaccess(s,o,read) \wedge permit(s,o,read) \rightarrow \bigcirc(permitaccess(s,o,read))$
$dominate(o.classfication, s.cleareance) \rightarrow permit(s,o,write)$
$tryaccess(s,o,write) \wedge permit(s,o,write) \rightarrow \bigcirc(permitaccess(s,o,write))$

- Example 3: DAC with ACL

$in((s.ID,r), o.acl) \rightarrow permit(s,o,r)$
$tryaccess(s,o,r) \wedge permit(s,o,r) \rightarrow \bigcirc(permitaccess(s,o,r))$

Specification of Core Models

- *onA*$_0$:

$\Box(\neg(p_1 \wedge ... \wedge p_i) \wedge (state(s,o,r) = accessing) \rightarrow \bigcirc(revokeaccess(s,o,r)))$

- Example 6:

$\Box(\neg(Bob.role = employee) \wedge (Bob.temp\_cert \in RCL)) \wedge (state(Bob,o,r) = accessing) \rightarrow \bigcirc(revokeaccess(Bob,o,r)))$

Specification of Core Models

- *preA*$_1$:

$p_1 \wedge ... \wedge p_i \rightarrow permit(s,o,r)$
$permitaccess(s,o,r) \rightarrow \blacklozenge(tryaccess(s,o,r) \wedge permit(s,o,r) \wedge \Diamond(preupdate(attribute)))$

- Example 4: DRM pay-per-use application

$(Alice.credit \geq ebook1.value) \rightarrow permit(Alice, ebook1, read)$
$permitaccess(Alice, ebook1, read) \rightarrow \blacklozenge(tryaccess(Alice, ebook1, read) \wedge \Diamond(preupdate(Alice.credit))) \wedge permit(Alice, ebook1, read)$
$preupdate : Alice.credit' = Alice.credit - ebook1.value$

Specification of Core Models

- *onA*$_1$:

$permitaccess(s,o,r \rightarrow \blacklozenge(tryaccess(s,o,r) \wedge \Diamond(preupdate(attribute)))$
$\Box(\neg(p_1 \wedge ... \wedge p_i) \wedge (state(s,o,r) = accessing) \rightarrow \bigcirc(revokeaccess(s,o,r)))$

- *onA*$_2$:

$\Box(\neg(p_1 \wedge ... \wedge p_i) \wedge (state(s,o,r) = accessing) \rightarrow \bigcirc(revokeaccess(s,o,r)))$
$endaccess(s,o,r) \vee revokeaccess(s,o,r) \rightarrow \blacklozenge(permitaccess(s,o,r) \wedge \Diamond(onupdate(attribute)))$

- *onA*$_3$:

$\Box(\neg(p_1 \wedge ... \wedge p_i) \wedge (state(s,o,r) = accessing) \rightarrow \bigcirc(revokeaccess(s,o,r)))$
$endaccess(s,o,r) \vee revokeaccess(s,o,r) \rightarrow \Diamond(postupdate(attribute))$

Specification: an Example

- Example 7: Resource-constrained access control
  - Limited number (10) of ongoing accessing for a single object
  - Object attribute: $accessingS = \{s | s \text{ is accessing } o\}$
  - When 11th subject requesting new access, one ongoing accessing subject will be revoked.
- a. revocation by earliest usage will be revoked
  - Subject attribute: *startTime*

(1) **true** $\rightarrow permit(s, o, r)$
(2) $permitaccess(s, o, r) \rightarrow \blacklozenge(preupdate(o.accessingS))$, where $preupdate : o.accessingS' = o.accessingS + \{s\}$
(3) $tryaccess(x, o, r) \wedge (x \notin o.accessingS) \wedge (|o.accessingS| = 10) \wedge (s \in o.accessingS) \wedge (s.startTime = Min(o.accessingS)) \rightarrow \bigcirc(revokeaccess(s, o, r))$
(4) $endaccess(s, o, r) \vee revokeaccess(s, o, r) \rightarrow \Diamond(postUpdate(o.accessingS)$, where $postUpdate : o.accessingS' = o.accessingS - \{s\}$

Specification: an Example

- b. revocation by longest idle usage
  - Subject attribute: *idleTime*

(1) **true** $\rightarrow permit(s, o, r)$
(2) $permitaccess(s, o, r) \rightarrow \blacklozenge(preupdate(o.accessingS))$, where $preupdate : o.accessingS' = o.accessingS + \{s\}$
(3) $tryaccess(x, o, r) \wedge (x \notin o.accessingS) \wedge (|o.accessingS| = 10) \wedge (s \in o.accessingS) \wedge (s.idleTime = Max(o.accessingS)) \rightarrow \bigcirc(revokeaccess(s, o, r))$

- c. revocation by longest total usage
  - Subject attribute: *usageTime*

(1) **true** $\rightarrow permit(s, o, r)$
(2) $permitaccess(s, o, r) \rightarrow \blacklozenge(preupdate(o.accessingS))$, where $preupdate : o.accessingS' = o.accessingS + \{s\}$
(3) $tryaccess(x, o, r) \wedge (x \notin o.accessingS) \wedge (|o.accessingS| = 10) \wedge (s \in o.accessingS) \wedge (s.usageTime = Max(o.accessingS)) \rightarrow \bigcirc(revokeaccess(s, o, r))$
(4) $endaccess(s, o, r) \vee revokeaccess(s, o, r) \rightarrow \Diamond(postupdate(usageTime) \wedge \Diamond(postupdate(accesingS))$, where $postupdate : o.accessingS' = o.accessingS - \{s\}$

Outline

- Introduction of UCON
- Temporal Logic of Action (TLA)
- Logic Model for UCON with TLA
- Specification of Authorization Core Models
- Obligation and Conditions
- Conclusions and Future Work

Obligations

- Two types of obligations in UCON:
  - pre-obligations, which must have been performed before access.
  - ongoing-obligations, which must be performed during usage.

Definition 3 *An obligation is an action described by:*

$$a_b(s, o, r, s_b, o_b, r_b, para_1, ..., para_i, ...)$$

where a_b is the obligation name, (s, o, r) is a particular usage process requiring the obligation, s_b, o_b, r_b are obligation subject, object and right, $para_1$, ..., $para_i$ are optional parameters .

Definition 4 *A logical model of UCON with authorizations and obligations is a 4-tuple:*

$$\mathcal{M} = (\mathcal{S}, \mathcal{P}, \mathcal{A}_\mathcal{A}, \mathcal{A}_\mathcal{B})$$

where $\mathcal{S}$ is a sequence of states, $\mathcal{P}$ is a finite set of predicates, $\mathcal{A}_\mathcal{A}$ is a finite set of authorization actions (same as the $\mathcal{A}$ in the authorization model), $\mathcal{A}_\mathcal{B}$ is a finite set of obligation actions.

Obligations

- Example: click license agreement before making order:

$$(s.role = registered) \rightarrow permit(s, o, order)$$
$$permit(s, o.order) \wedge \ominus(click\_agreement(s, o, order, s, agree\_statement, click)) \rightarrow permitaccess(s, o, order)$$

Conditions

- Conditions are environment restrictions before or during usage.
- In UCON, a condition is a predicate built from system attributes, such as time and location.

Definition 5 *A logical model of UCON with authorizations, obligations, and conditions is a 5-tuple:*

$$\mathcal{M} = (\mathcal{S}, \mathcal{P}_\mathcal{A}, \mathcal{P}_\mathcal{C}, \mathcal{A}_\mathcal{A}, \mathcal{A}_\mathcal{B})$$

where $\mathcal{S}$, $\mathcal{A}_\mathcal{A}$, and $\mathcal{A}_\mathcal{B}$ are the same as before, $\mathcal{P}_\mathcal{A}$ is a finite set of authorization predicates (the $\mathcal{P}$ before), and $\mathcal{P}_\mathcal{C}$ is a finite set of condition predicates.

- Example:

$$(s.role = dayshifter) \wedge (8am \leq currentT \leq 5pm) \rightarrow permitaccess(s, o, r)$$
$$(s.role = nightshifter) \wedge \neg(8am \leq currentT \leq 5pm) \rightarrow permitaccess(s, o, r)$$

Outline

- Introduction of UCON
- Temporal Logic of Action (TLA)
- Logic Model for UCON with TLA
- Specification of Authorization Core Models
- Obligation and Conditions
- Conclusions and Future Work

Conclusions

- A logical model for UCON with:
  - States with:
    - subject attributes and values
    - Object attributes and values
    - System attribute and values
  - Predicates:
    - Authorization predicates built from subject and object attributes
    - Condition predicates built from system attributes
  - Actions:
    - Attribute update actions
    - Usage control actions
    - Obligation actions
  - Temporal formulas of usage control policies
- First-order logic specification of the UCON models with new features of:
  - Mutability
  - Continuality

Future Work

- UCON:
  - Enrich UCON model, such as constraints, delegations
  - Administrative UCON model
    - Attribute management
    - Administrative policies
  - Expressive power and safety analysis for UCON
  - Concurrency of UCON
- Development of architecture and mechanism for UCON system