Safety Analysis Of Attribute Based Access Control Model(ABACα)

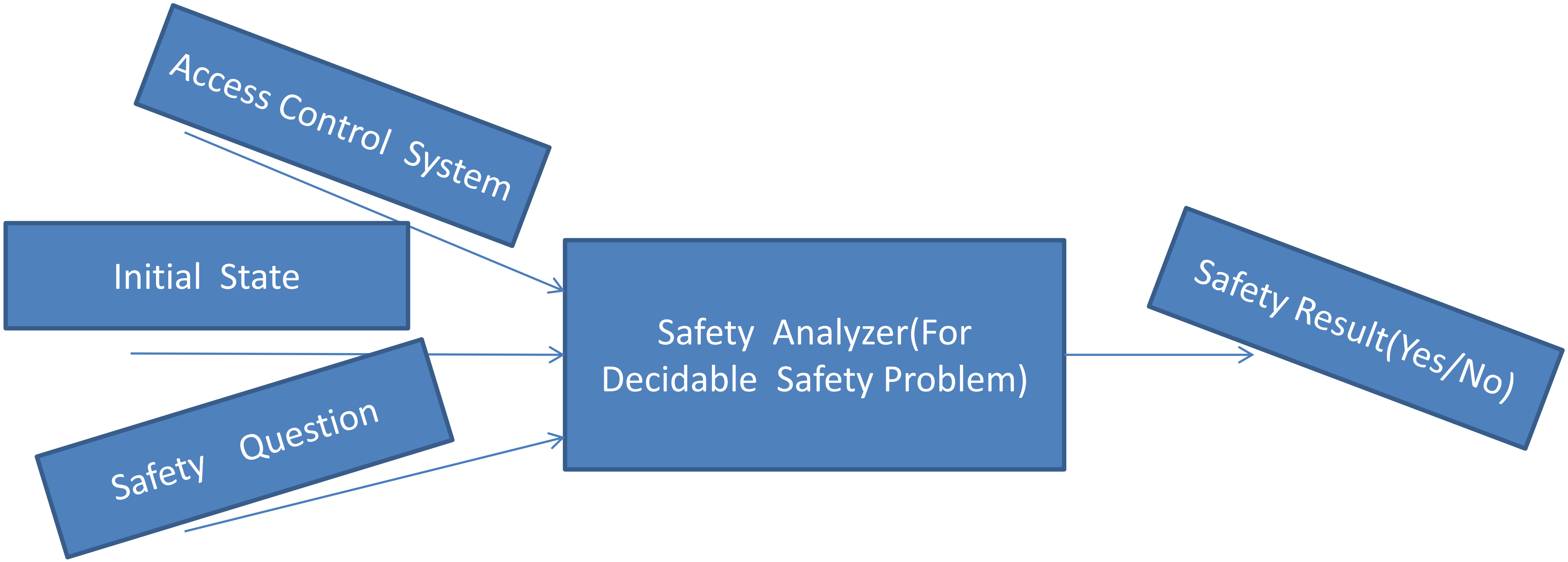
Tahmina Ahmed and Ravi Sandhu

What is Safety in an Access Control Model?

Definition 1. *Safety Of an Access Control Model: Given a system Of an access control model and an initial state of the system a safety question asks whether a subject can have certain rights on an object after execution of a sequence of authorized commands*

Example For HRU

Safety Question in given a HRU system, an initial state $\langle S^0, O^0, M^0[], R\rangle$ where S^0, O^0 are set of subjects, objects from initial state and $M^0[]$ access matrix of initial state ,R is the set of Rights, a subject $s \in S^0$, an object $o \in O^0$ and a right $r \in R$ whether in any state γ $r \in M^{\gamma}[s,o]$.



ABACα Model

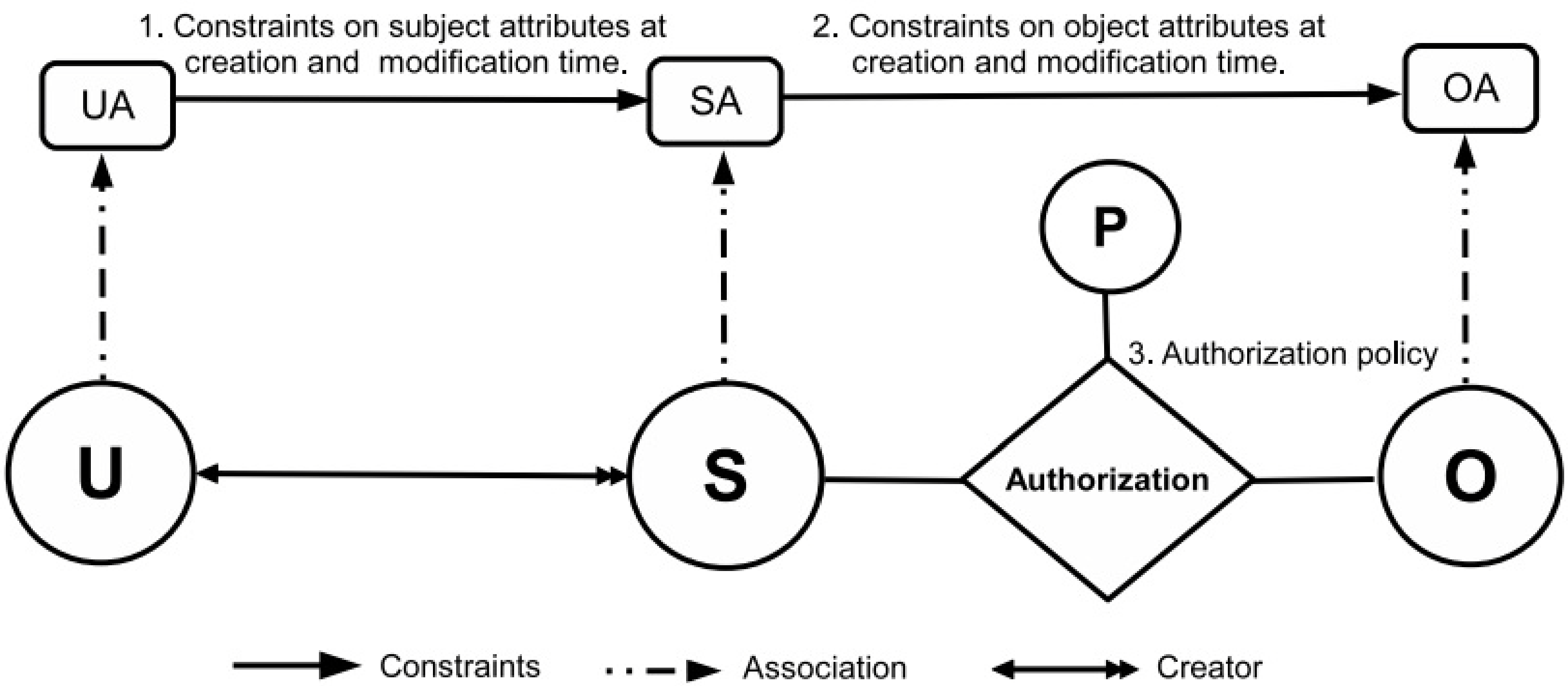


Fig. 1. *Unified ABAC model structure.*

Basic Sets and Functions of ABACα

U, S and O represent finite sets of *existing* users, subjects and objects respectively.

UA, SA and OA represent finite sets of user, subject and object attribute functions respectively. (Henceforth referred to as simply attributes.)

P represents a finite set of permissions.

For each *att* in UA ∪ SA ∪ OA, Range(*att*) represents the attribute's range, a finite set of *atomic* values.

SubCreator: $S \rightarrow U$. For each subject SubCreator gives its creator.

attType: UA ∪ SA ∪ OA → {set, atomic}. Specifies attributes as set or atomic valued.

Each attribute function maps elements in U, S and O to atomic or set values.

$$\forall ua \in \mathrm{UA}.\ ua : \mathrm{U} \rightarrow \begin{cases} \mathrm{Range(ua)} & \text{if } \mathrm{attType}(ua) = \mathrm{atomic} \\ 2^{\mathrm{Range(ua)}} & \text{if } \mathrm{attType}(ua) = \mathrm{set} \end{cases}$$

$$\forall sa \in \mathrm{SA}.\ sa : \mathrm{S} \rightarrow \begin{cases} \mathrm{Range(sa)} & \text{if } \mathrm{attType}(sa) = \mathrm{atomic} \\ 2^{\mathrm{Range(sa)}} & \text{if } \mathrm{attType}(sa) = \mathrm{set} \end{cases}$$

Safety Question On ABACα

Let the Initial state Users subjects objects can be represented as U^0, S^0, O^0. And $\mathcal{P}(\mathcal{S})$ is the Power Set of set S.

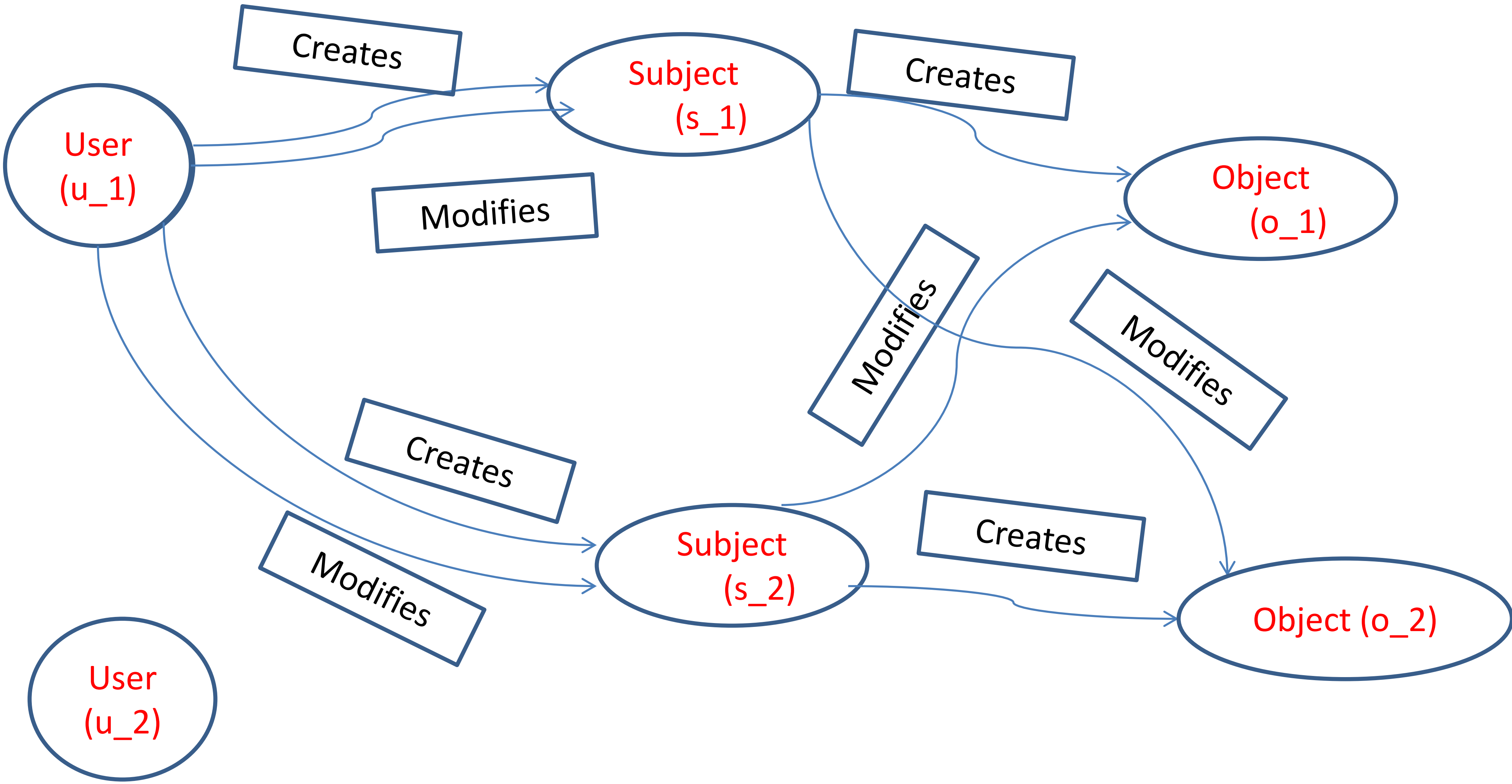
Definition 2. *Specific Object Safety Question: Given an object $o \in O^0$ an attribute $oa \in OA$ and a value v where $v \in \text{Range}(oa)$ if $\text{attType}(oa)$=atomic or $v \in \mathcal{P}(\text{Range}(oa))$ if $\text{attType}(oa)$=set whether the system can reach in a state where $oa(o)=v$?*

Definition 3. *Specific Subject Safety Question: Given a subject $s \in S^0$ an attribute $sa \in SA$ and a value v where $v \in \text{Range}(sa)$ if $\text{attType}(sa)$=atomic or $v \in \mathcal{P}(\text{Range}(sa))$ if $\text{attType}(sa)$=set whether the system can reach in a state where $sa(s)=v$?*

Safety Analysis Scope of ABACα

- User Creation, Modification and Deletion are Administrative operations of and out of scope for this analysis.
- Our assumption is that in respect of safety analysis consideration of subject deletion wont make the system more vulnerable comparing to the system which disregard deletion. So we haven't consider Subject deletion in our analysis.

Our Analysis Scope



Is Safety Of ABACα UnDecidable?

To answer this question we need to give a try to reduce $ABAC_\alpha$ to a standard undecidable problem. We have tried with the following two problems

- Halting Problem of Turing machine.
- Post Correspondence Problem

Turing Machine

A general Turing machine with one dimensional single tape $\mathcal{M}$ is a 6 tuple:{Q,Σ,δ,q_0, q_{accept}, q_{reject} }, where:

- Q is a finite set of states,
- Σ is a finite set, alphabet with *blank*
- $\delta : Q \times \Sigma \longrightarrow Q \times \Sigma \times \{L, R\}$ is the transition function,
- $q_0, q_{accept}, q_{reject} \in Q$ are the start state ,accept state,and reject state, respectively, where $q_{accept} \neq q_{reject}$

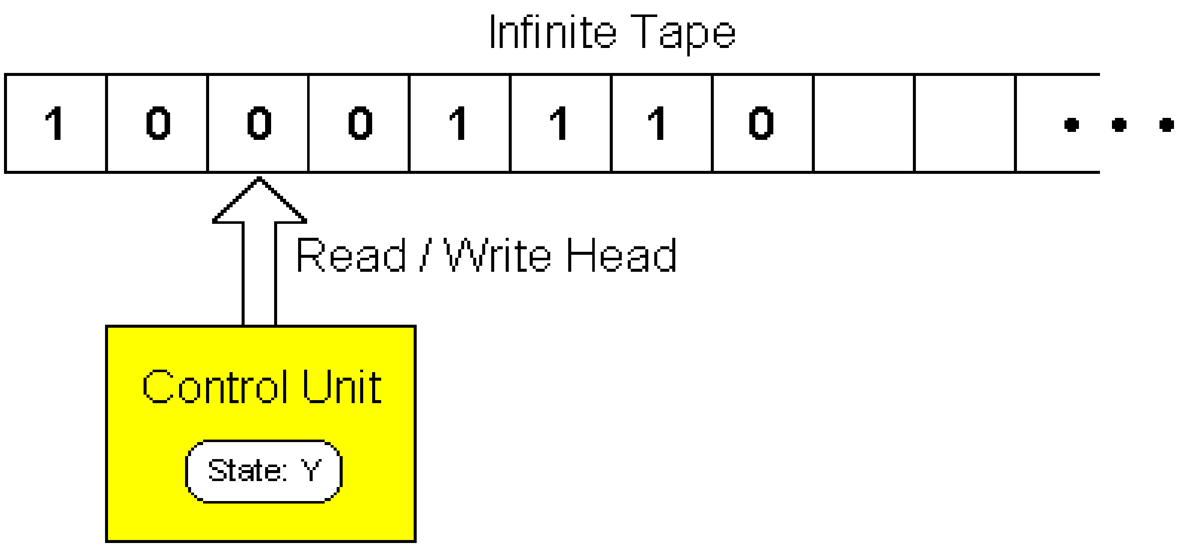
Turing Machine

The movement of the head in the tape in described as below:

- $\delta(q, x)$ = (p,y,L) in state q the tape head searching for the cell containing x and the head write y on that cell moves one cell to the left on the tape and the new state should be named as p. If the tape head is at the left end no movement will occur. Left transition can be of two types:
  1. Left transition when head pointing to the left end as it is a one way tape no creation will occur resulting this transition. Only cell content and state will change
  2. Left transition when head not pointing to the left end
- $\delta(q, x)$ = (p,y,R) same as above only moves right.

  Right transition can be of two types:
  1. Right transition when head pointing to the right end new cell should be created to move the head right.
  2. Right transition when head not pointing to the right end.

Turing Machine



http://turing.slc.edu/~jmarshall/courses/2002/fall/cs30/Lectures/week08/Computation.html

Extension of ABACα Require to Construct TM

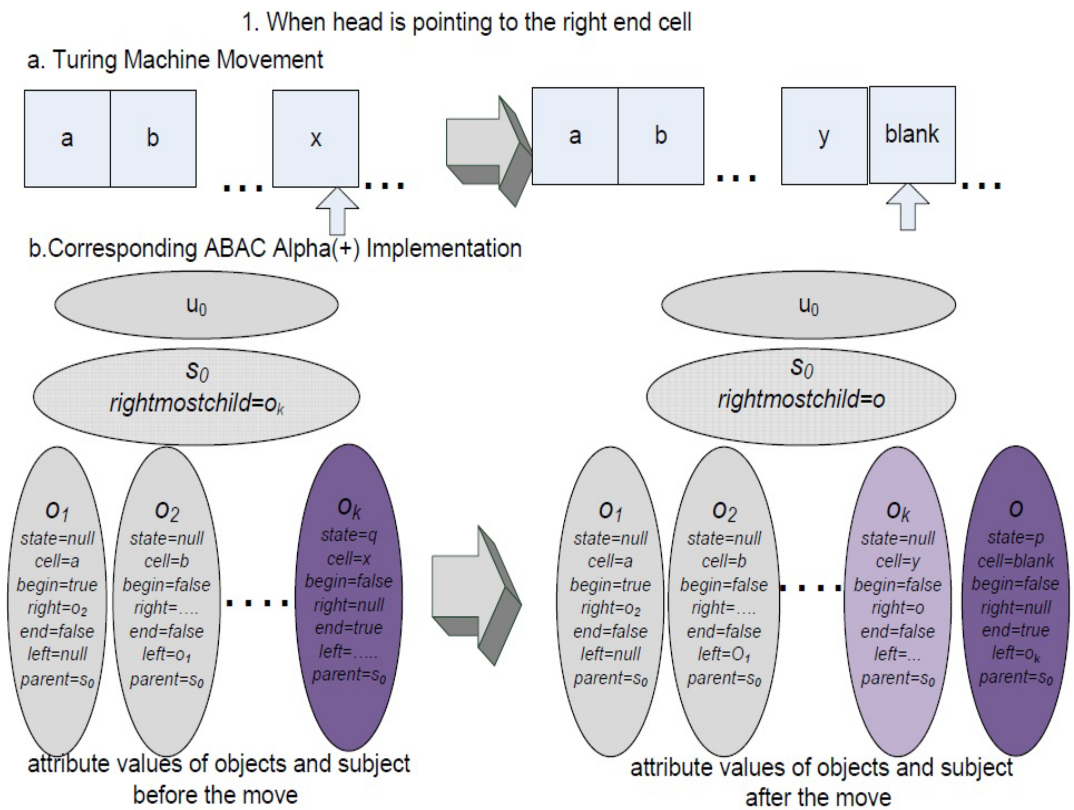
To construct one way single tape Turing Machine with ABAC_α

- Infinite attribute range for subjects and objects attribute.
- Ability to modify linked attribute. For example $oa_1 \in$ OA, $oa_2 \in$ OA and Range (oa)=O, Let $oa_1(o_1) = o_2$. During modification of o_1 the function can access $oa_2(oa_1(o1))$ which is similar to $oa_2(o_2)$.
- Ability to modify subject attribute during object creation.

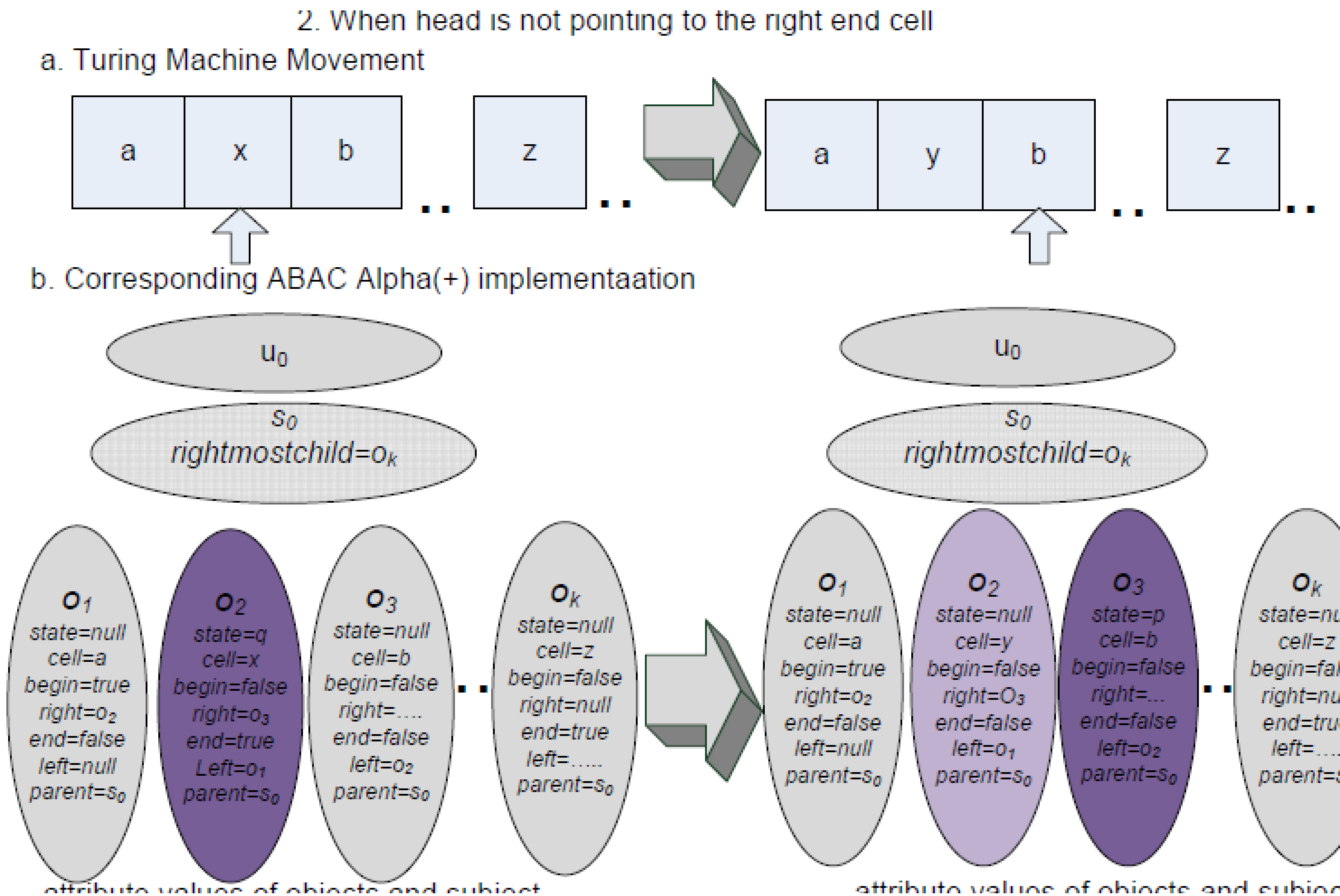
ABAC_α doesn't supports any of the above functionality so without these extensions it's not possible to simulate the one way single tape Turing machine.

So other than the extension of ABAC_α attribute range we cannot simulate PCP with ABAC_α.

Construction Of Turing Machine With ABACα (+)



Simulation of Right Move



PCP Construction

The input of the problem consists of two finite lists $\alpha_1, \ldots, \alpha_N$ and $\beta_1, \ldots, \beta_N$ of words over some alphabet A having at least two symbols. A solution to this problem is a sequence of indices $(i_k)_{1 \le k \le K}$ with $K \ge 1$ and $1 \le i_k \le N$ for all k, such that

$$\alpha_{i_1} \ldots \alpha_{i_K} = \beta_{i_1} \ldots \beta_{i_K}.$$

The decision problem then is to decide whether such a solution exists or not.

α_1	α_2	α_3
a	ab	bba

β_1	β_2	β_3
baa	aa	bb

A solution to this problem would be the sequence (3, 2, 3, 1), because

$$\alpha_3\alpha_2\alpha_3\alpha_1 = bba + ab + bba + a = bbaabbbaa = bb + aa + bb + baa = \beta_3\beta_2\beta_3\beta_1.$$

Furthermore, since (3, 2, 3, 1) is a solution, so are all of its "repetitions", such as (3, 2, 3, 1, 3, 2, 3, 1), etc.; that is, when a solution exists, there are infinitely many solutions of this repetitive kind.

However, if the two lists had consisted of only α_2, α_3 and β_2, β_3 from those sets, then there would have been no solution (the last letter of any such α string is not the same as the letter before it, whereas β only constructs pairs of the same letter).

A convenient way to view an instance of a Post correspondence problem is as a collection of blocks of the form

α_i
β_i

Source: http://en.wikipedia.org/wiki/Post_correspondence_problem

PCP Construction with ABACα

PCP Construction needs the

- Range of the attribute value should be infinite
- its type should be string .

$ABAC_\alpha$ doesn't supports attribute with infinite range. Range of every attribute a∈ UA∪ SA∪ OAis finite.

Is Safety Of ABAC Decidable?

For Convenience of Analysis we have defined some notation

Let $\mathcal{P}(S)$ denote the power set of S. Note that each element in $\mathcal{P}(S)$ is a set.

$$\text{For each } ua \text{ in UA}, \text{UASET}_{ua} = \begin{cases} \{ua\} \times \text{Range (ua)} & \text{if attType } (ua) = \text{atomic} \\ \{ua\} \times 2^{\text{Range (ua)}} & \text{if attType } (ua) = \text{set} \end{cases}$$

$$\text{For each } sa \text{ in SA}, \text{SASET}_{sa} = \begin{cases} \{ua\} \times \text{Range (sa)} & \text{if attType } (sa) = \text{atomic} \\ \{ua\} \times 2^{\text{Range (sa)}} & \text{if attType } (sa) = \text{set} \end{cases}$$

$$\text{For each } sa \text{ in SA}, \text{SASET}_{sa} = \begin{cases} \{ua\} \times \text{Range (sa)} & \text{if attType } (sa) = \text{atomic} \\ \{ua\} \times 2^{\text{Range (sa)}} & \text{if attType } (sa) = \text{set} \end{cases}$$

$$\text{For each } oa \text{ in OA}, \text{OASET}_{oa} = \begin{cases} \{oa\} \times \text{Range (oa)} & \text{if attType } (oa) = \text{atomic} \\ \{oa\} \times 2^{\text{Range (oa)}} & \text{if attType } (oa) = \text{set} \end{cases}$$

Let $\text{POW}_{ua} = \mathcal{P}(\bigcup_{\forall ua \in \text{UA}} \text{UASET}_{ua})$

$$UASET = \{x | x \in \text{POW}_{ua} \wedge |x| = |\text{UA}| \wedge \\ \forall a \in \text{UA}.\forall val1, val2 \in \text{Range}(a). \\ \langle a, val1 \rangle \in x \wedge (val1 \neq val2) \rightarrow \langle a, val2 \rangle \notin x\}$$

Let $\text{POW}_{sa} = \mathcal{P}(\bigcup_{\forall sa \in \text{SA}} \text{SASET}_{sa})$

$$SASET = \{x | x \in \text{POW}_{sa} \wedge |x| = |\text{SA}| \wedge \\ \forall a \in \text{SA}.\forall val1, val2 \in \text{Range}(a). \\ \langle a, val1 \rangle \in x \wedge (val1 \neq val2) \rightarrow \langle a, val2 \rangle \notin x\}$$

Let $\text{POW}_{oa} = \mathcal{P}(\bigcup_{\forall oa \in \text{OA}} \text{OASET}_{oa})$

$$OASET = \{x | x \in \text{POW}_{oa} \wedge |x| = |\text{OA}| \wedge \\ \forall a \in \text{OA}.\forall val1, val2 \in \text{Range}(a). \\ \langle a, val1 \rangle \in x \wedge (val1 \neq val2) \rightarrow \langle a, val2 \rangle \notin x\}$$

Example

Example of UASET

let UA = $\{ua_1, ua_2\}$

Range(ua_1)={1,2,3}

attType(ua_1)= atomic

Range(ua_2) = { a, b, c}

attType = set

$UASET_{ua_1}$ ={1, 2 , 3 }

$UASET_{ua_2}$ ={{},{a},{b},{c},{a,b},{a,c},{b,c}, {a,b,c}}

UASET = $\{\langle 1, \{\}\rangle, \{1, \{a\}\}, \ldots\ldots\}$

Notations

Q^{γ}_{users}=AllPossibleSatesOfUsers(U^{γ}, UASET)

$$\text{AllPossibleSatesOfUsers}(\ U^{\gamma},\ \text{UASET})$$
$$= \{x | x \in \mathcal{P}(U^{\gamma} \times \text{UASET}) \wedge |x| = |U^{\gamma}| \wedge$$
$$\forall u \in U^{\gamma}.\forall uaset_1, uaset_2 \in \text{UASET}.$$
$$\langle u, uaset_1 \rangle \in x \wedge (uaset_1 \neq uaset_2) \rightarrow \langle u, uaset_2 \rangle \notin x\}$$

$Q^{\gamma}_{objects}$=AllPossibleSatesOfObjects(O^{γ}, OASET)

$$\text{AllPossibleSatesOfObjects}(\ O^{\gamma},\ \text{OASET})$$
$$= \{x | x \in \mathcal{P}(O^{\gamma} \times \text{OASET}) \wedge |x| = |O^{\gamma}| \wedge$$
$$\forall o \in O^{\gamma}.\forall oaset_1, oaset_2 \in \text{OASET}.$$
$$\langle o, oaset_1 \rangle \in x \wedge (oaset_1 \neq oaset_2) \rightarrow \langle o, oaset_2 \rangle \notin x\}$$

$Q^{\gamma}_{subjects}$=AllPossibleStatesOfSubjects(S^{γ},SASET) Where

$$\text{AllPossibleStatesOfSubjects}(S^{\gamma},\text{SASET})$$
$$= \{x | x \in \mathcal{P}(S^{\gamma} \times \text{SASET}) \wedge |x| = |S^{\gamma}| \wedge$$
$$\forall s \in S^{\gamma}.\forall saset_1, saset_2 \in \text{SASET}.$$
$$\langle s, saset_1 \rangle \in x \wedge (saset_1 \neq saset_2) \rightarrow \langle o, saset_2 \rangle \notin x\}$$

Example

EXAMPLE FOR Q^{γ}_{users}

Let $U^{\gamma} = \{u_1, u_2, u_3\}$

$\text{UASET} = \{uaset_1, uaset_2\}$

$$Q^{\gamma}_{users} = \{ \langle\langle u_1, uaset_1, \rangle, \langle u_2, uaset_1, \rangle, \langle u_3, uaset_1, \rangle\rangle,$$
$$\langle\langle u_1, uaset_1, \rangle, \langle u_2, uaset_2, \rangle, \langle u_3, uaset_1, \rangle\rangle,$$
$$\langle\langle u_1, uaset_1, \rangle, \langle u_2, uaset_1, \rangle, \langle u_3, uaset_2, \rangle\rangle,$$
$$\langle\langle u_1, uaset_2, \rangle, \langle u_2, uaset_2, \rangle, \langle u_3, uaset_2, \rangle\rangle,$$
$$\langle\langle u_1, uaset_2, \rangle, \langle u_2, uaset_1, \rangle, \langle u_3, uaset_2, \rangle\rangle,$$
$$\langle\langle u_1, uaset_2, \rangle, \langle u_2, uaset_1, \rangle, \langle u_3, uaset_1, \rangle\rangle\}$$

Notational Change From ABACα

Notational Change From $ABAC_\alpha$

- LConstrObj contains only the current and new attribute value of the subject and object so without loss of generality we can rewrite ConstrObj like as follows:
  ConstrObj ($saset^{curr}$:SASET,$oaset^{new}$:OASET)
- LConstrObjMod contains only the current and new attribute value of the subject and object so without loss of generality we can rewrite ConstrObj like as follows:
  ConstrObj ($saset^{curr}$:SASET,$oaset^{curr}$: OASET,$oaset^{new}$:OASET)
- LConstrSub contains only the current and new attribute value of the user and subject so without loss of generality we can rewrite ConstrSub like as follows:
  ConstrSub ($uaset^{curr}$:UASET,$saset^{new}$:SASET)
- LConstrSubMod contains only the current and new attribute value of the subject and object so without loss of generality we can rewrite ConstrObj like as follows:
  ConstrSubMod($uaset^{curr}$:UASET,SASET^{curr} : SASET,$saset^{new}$:SASET)

Observation Towards Decidability Analysis

- In ABACα Object creation and modification of newly created object's attribute doesn't have any impact on existing subjects' and objects' attribute value. So we can disregard newly created object during our analysis.
- Subject Creation and modification of any subject attribute value has an impact on object creation and modification in the system.
- Only the subject's attribute value not the subject itself is important for the analysis.
- We need to create as many subjects necessary and sufficient to figure out the worst case modification of object's attribute value.

  This will be called the unfolding state.

Operations on Unfolding State

- No subject or object creation operation
- Construct a finite state machine with unfolding state subject and initial state objects
- Transition functions are

Subject Modification: user can modify only initial state subjects

Object Modification: Any Subject From the unfolding

State can modify objects of initial state.

Subject Attribute Update Graph

Subject Attribute Update Graph For ABAC_α ($\text{SAUG}_{\text{ABAC}_\alpha}$)

Nodes: $\{\langle x, y \rangle \mid x \in \text{SASET} \wedge y \in \text{UASET}\}$

Edges: there is an edge from u to v where

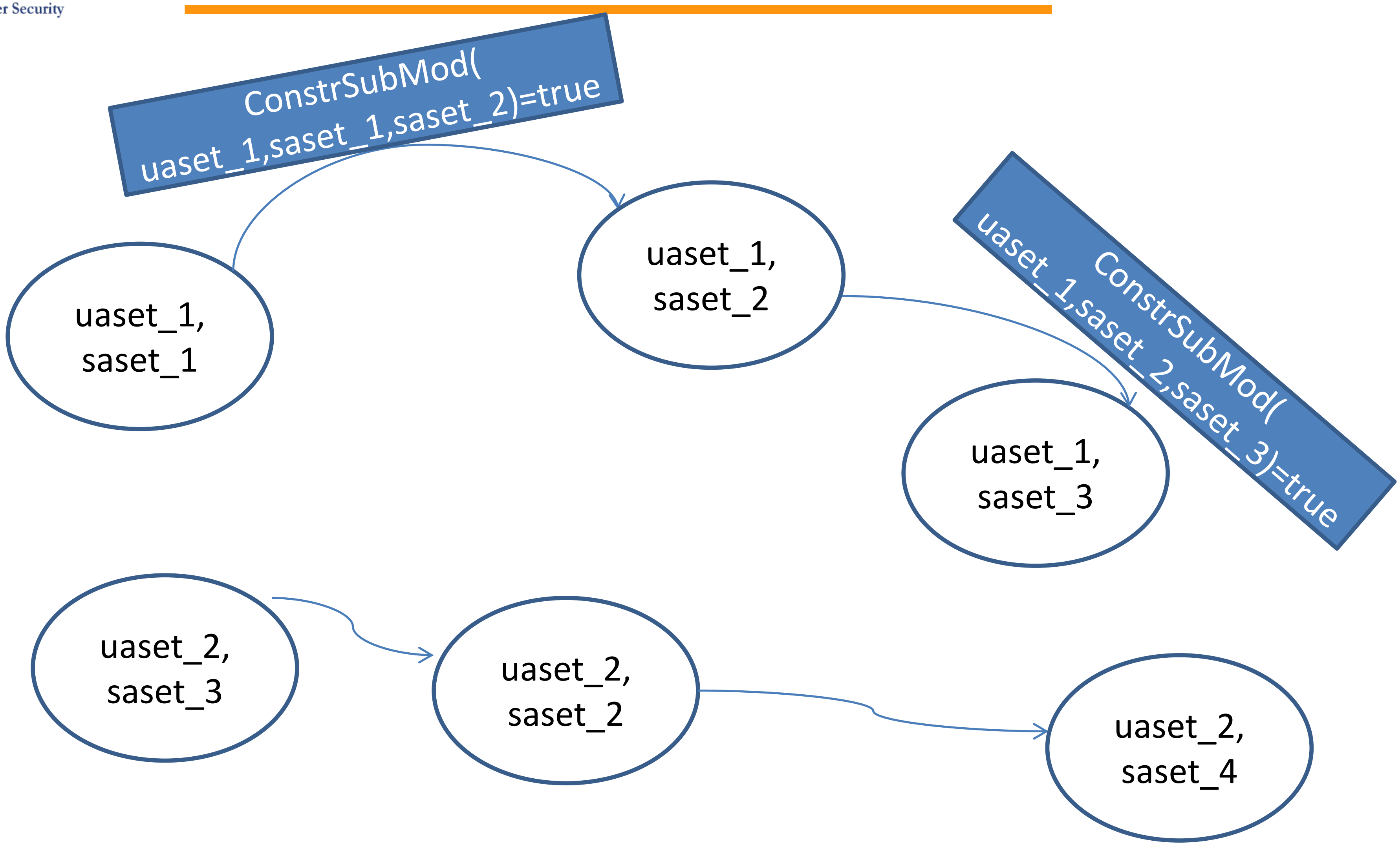
$u = \langle p, r \rangle$

and

$v = \langle q, r \rangle$

and $\text{ConstrSubMod}(r, p, q) = true$

Subject Attribute Update Graph



Subject Attribute Creation Set and Reachable Set

Subject Attribute Creation set(SACS) SACS is a set of tuples $\{ \langle u, saset \rangle \mid$ where $u \in \mathrm{U}^0 \wedge saset \in \mathrm{SASET}$ ConstrObj$(uaset(u), saset) = true\}$

Subject Attribute Reachable set(SARS) SARS is a set of tuples $\{ \langle u, saset_2 \rangle \mid$ where there exists an $saset_1$ such that $\langle u, saset_1 \rangle \in SACS \wedge saset_1 \neq saset_2 \wedge$ There is a PATH in $\mathrm{SAUG}_{\mathrm{ABAC}_\alpha}$ from $\langle uaset(u), saset_1 \rangle$ to $\langle uaset(u), saset_2 \rangle \}$

Subject Mapping Function

Definition 4. *The Subject Mapping Function* $\sigma_{mapping}$ *of* ABAC_α *scheme maps two subjects from any state h to the unfolding state when both are derived form same initial state. More Precisely for* ABAC_α *syatem, if h is any state and k is the unfolding state construction of the system both are derived from the same initial state 0 then the*

Subject Mapping Function $\sigma^{h,k}_{mapping} : S^h \mapsto S^k$

That is $\sigma^{h,k}_{mapping}(X) \equiv \{\ Y \in S^k \mid (X \in S^0 \implies X = Y) \lor (X \notin S^0 \implies (saset^h(X) = saset^k(Y) \land SubCreator(X) = SubCreator(Y) \land Y \notin S^0\ \}$

Unfolding Algorithm

Let the set of Subjects being created during the construction of the unfolding state is S^{USC}.

UnFolding Algorithm

i=0
$\mathrm{S}^{USC} = \emptyset$
$q^{USC}_{subjects} = \emptyset$
for each $\langle u, saset_1, saset_2\rangle \in SARS$
$\quad \mathrm{S}^{USC} = \mathrm{S}^{USC} \cup \mathrm{s}_i$
$\quad$ for each $\langle sa, val\rangle \in saset_2$
$\quad\quad sa^{US}(s_i) = \mathrm{val}$
$\quad q^{USC}_{subjects} = q^{USC}_{subjects} \cup \langle s_i, saset_2\rangle$
$\quad i++$

$\mathrm{S}^{US} = (\mathrm{S})^0 \cup \mathrm{S}^{USC}$
$q^{US}_{subjects} = (q_{subjects})^0 \cup q^{USC}_{subjects}$

runtime for the construction of unfolding state: $\mathrm{U}^0 \times \mathrm{SASET}$
Where S^0 = Set Of All subjects from initial state
$q^0_{subjects}$ = set of all possible subject SASET tuple

Runtime Analysis

RUNTIME ANALYSIS

Unfolding state Creation :$(\mid U^0 \mid \times \mid SASET \mid)$

Finite State Machine States :

$((\mid S^0 \mid \times \mid SASET \mid) + \mid U^0 \times SASET \mid) \times (\mid O^0 \mid \times \mid OASET \mid)$


Thank you