Evil twins:
Handling repetitions in attack–defense trees

A survival guide

Angèle Bossuat and **Barbara Kordy**

GraMSec 2017

Outline

Definition

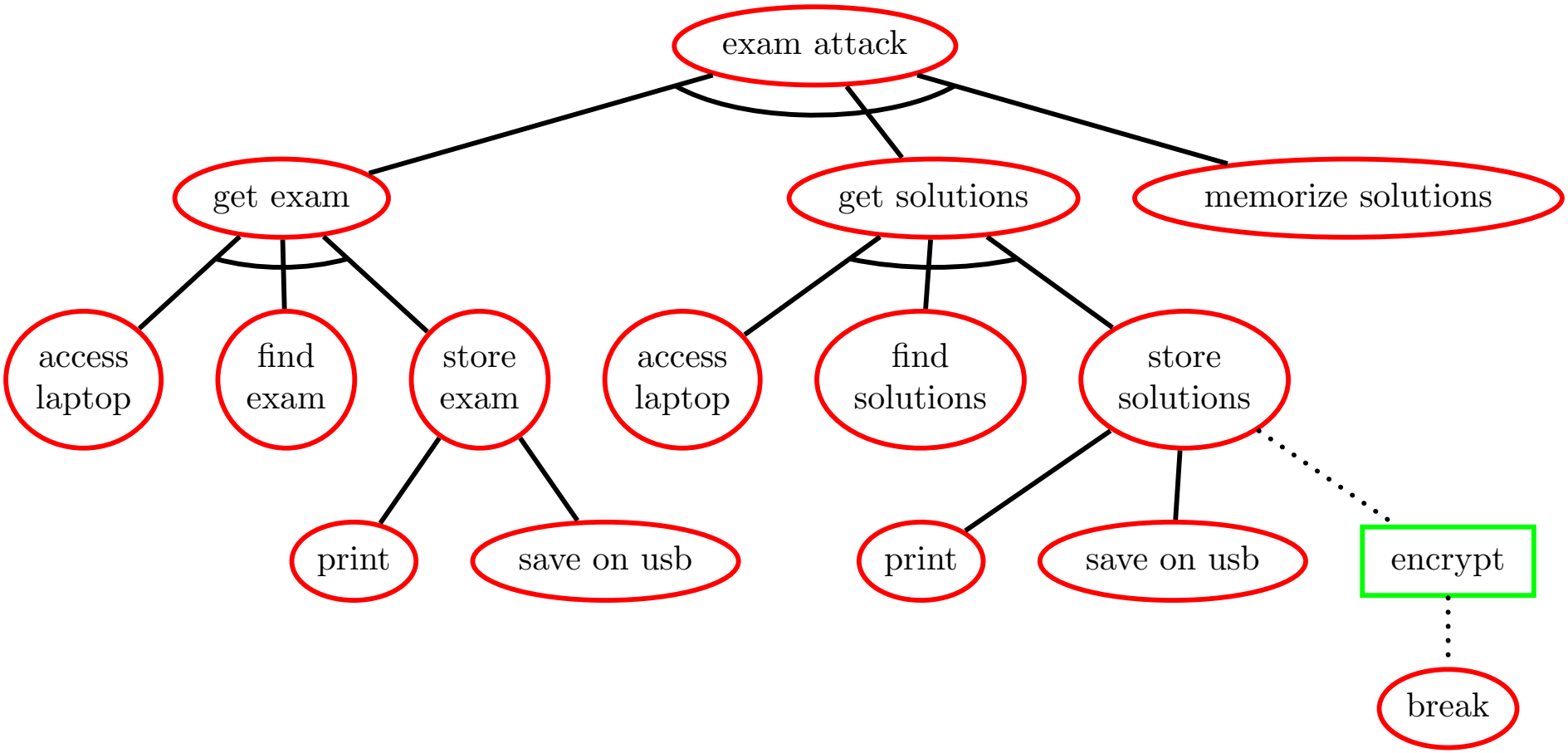
Attack–defense tree (ADTree)

A tree-like representation of a security scenario involving two actors: an attacker and a defender

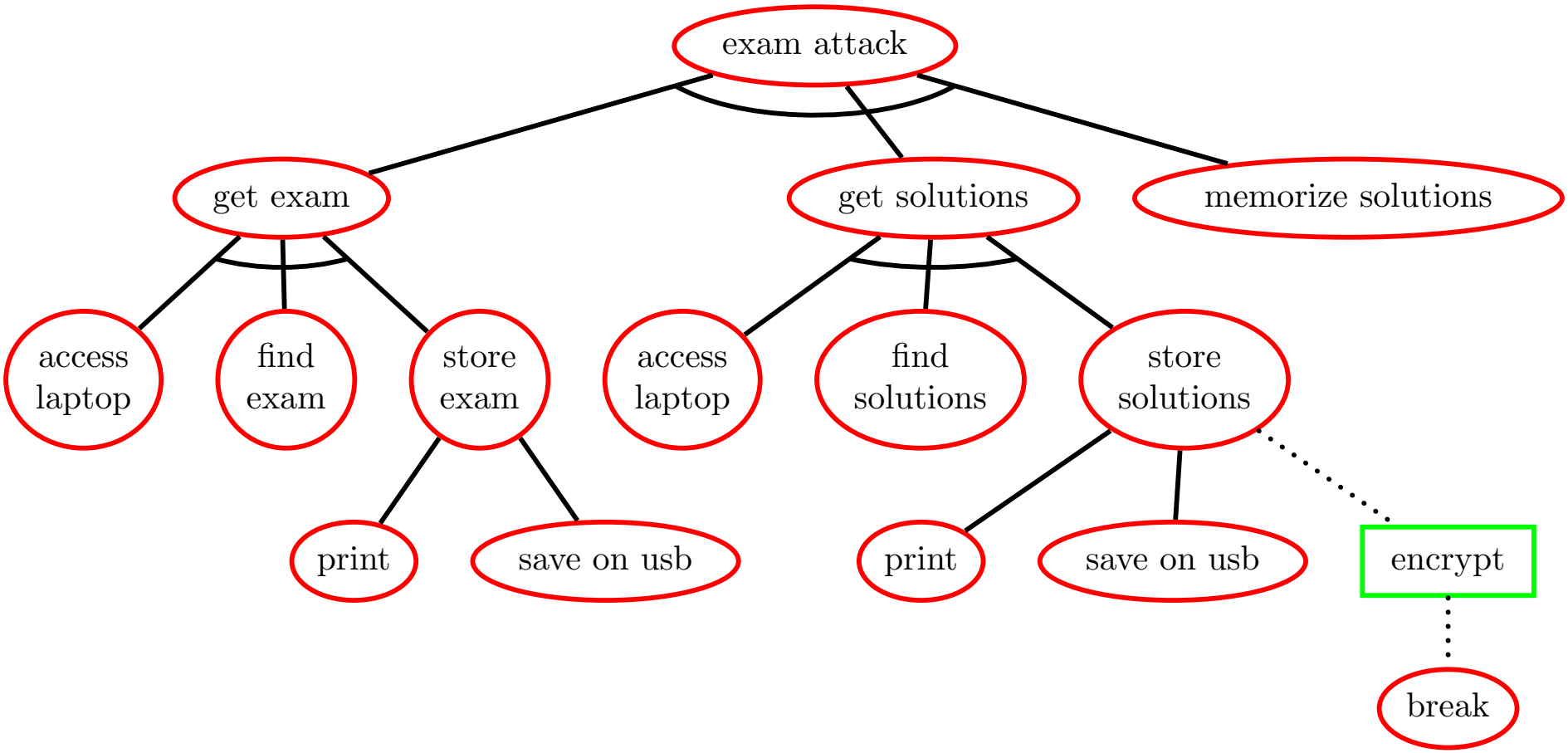
- Nodes represent the actors' goals
- Goals can be refined disjunctively (`OR`) or conjunctively (`AND`)
- Goals of one actor counter the goals of the other one

ADTrees extend classical attack trees with the nodes of the defender whose goal is to protect the modeled system

ADTree for passing the examination

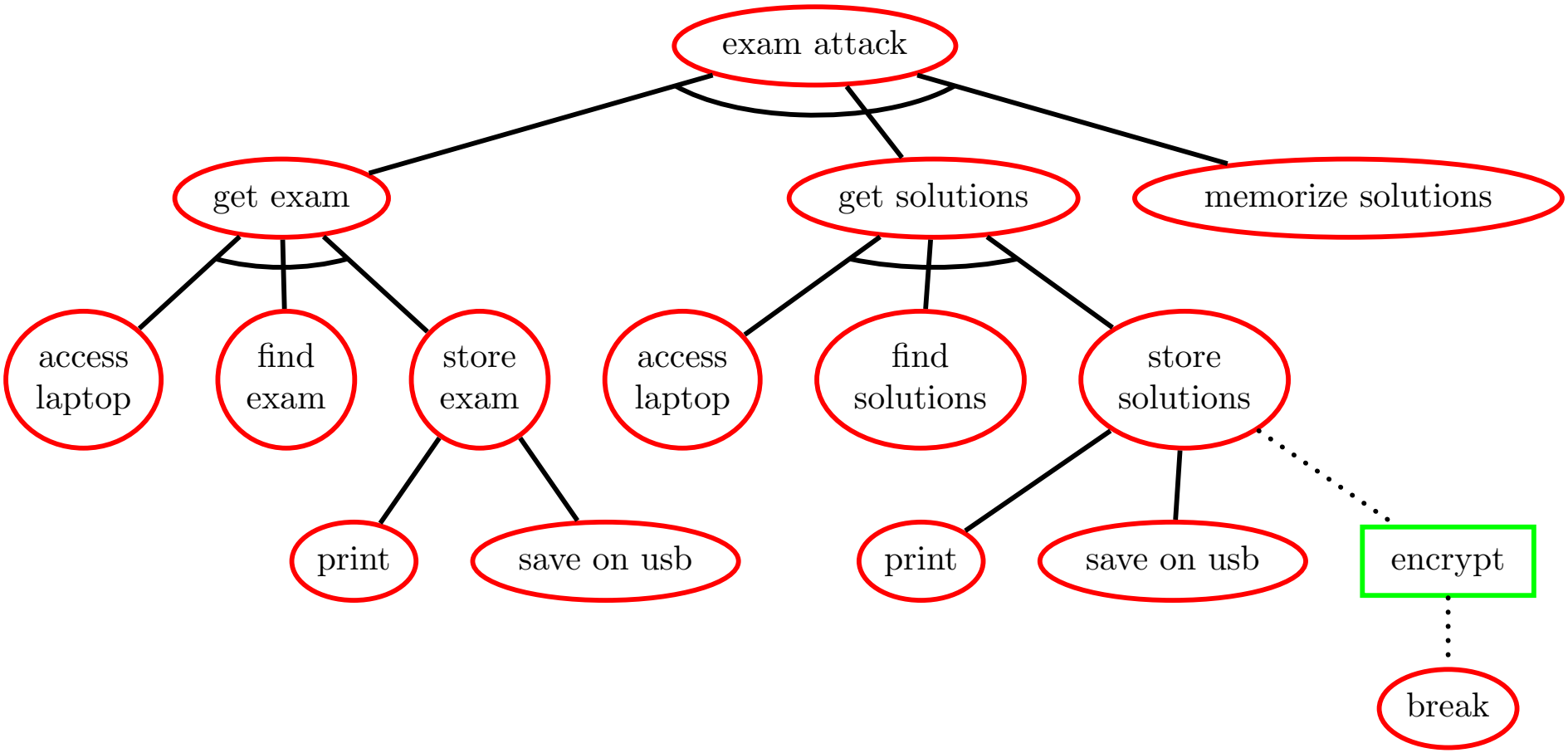


ADTree for passing the examination



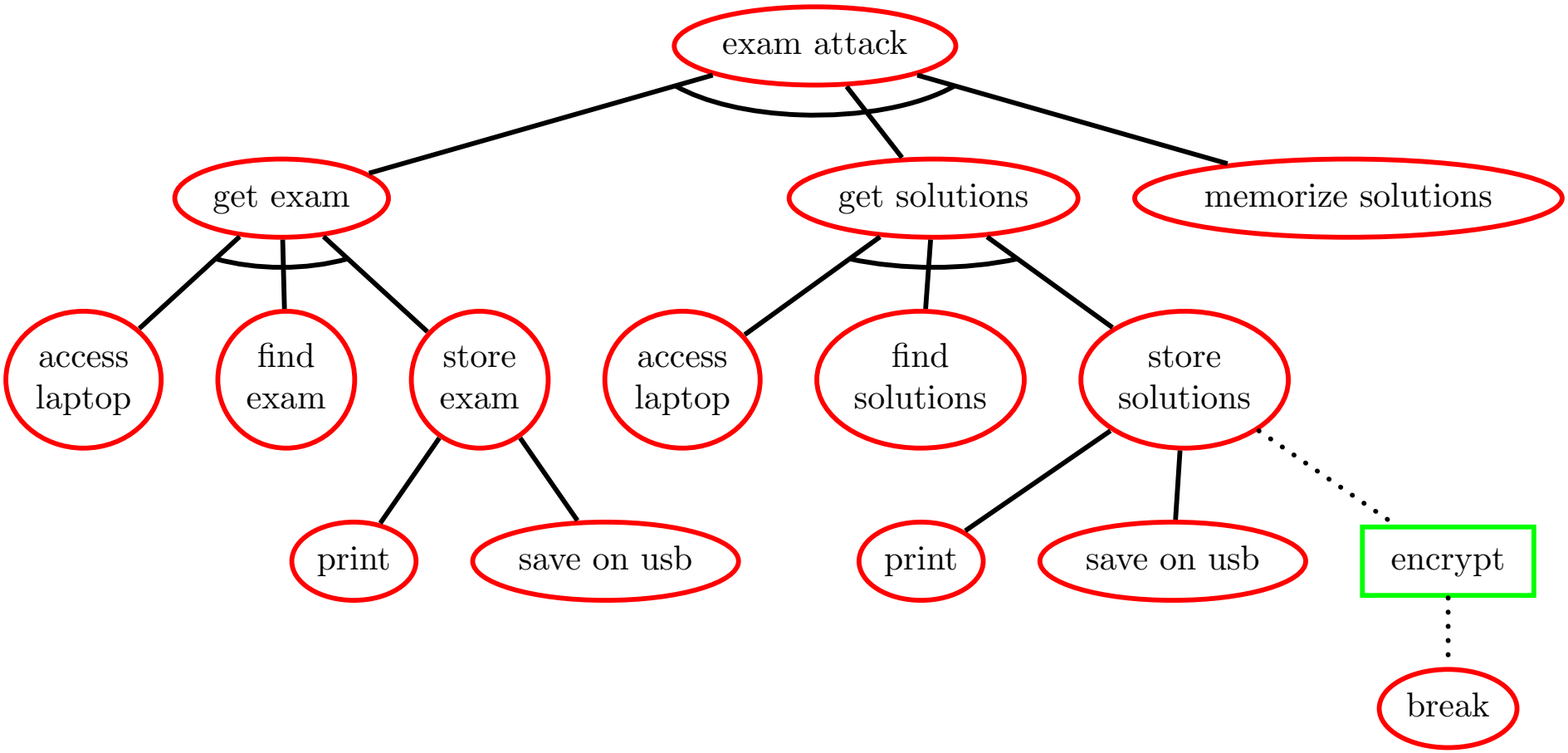
Refinement

ADTree for passing the examination



Countermeasures

ADTree for passing the examination



Basic actions = non-refined nodes

ADTrees as terms

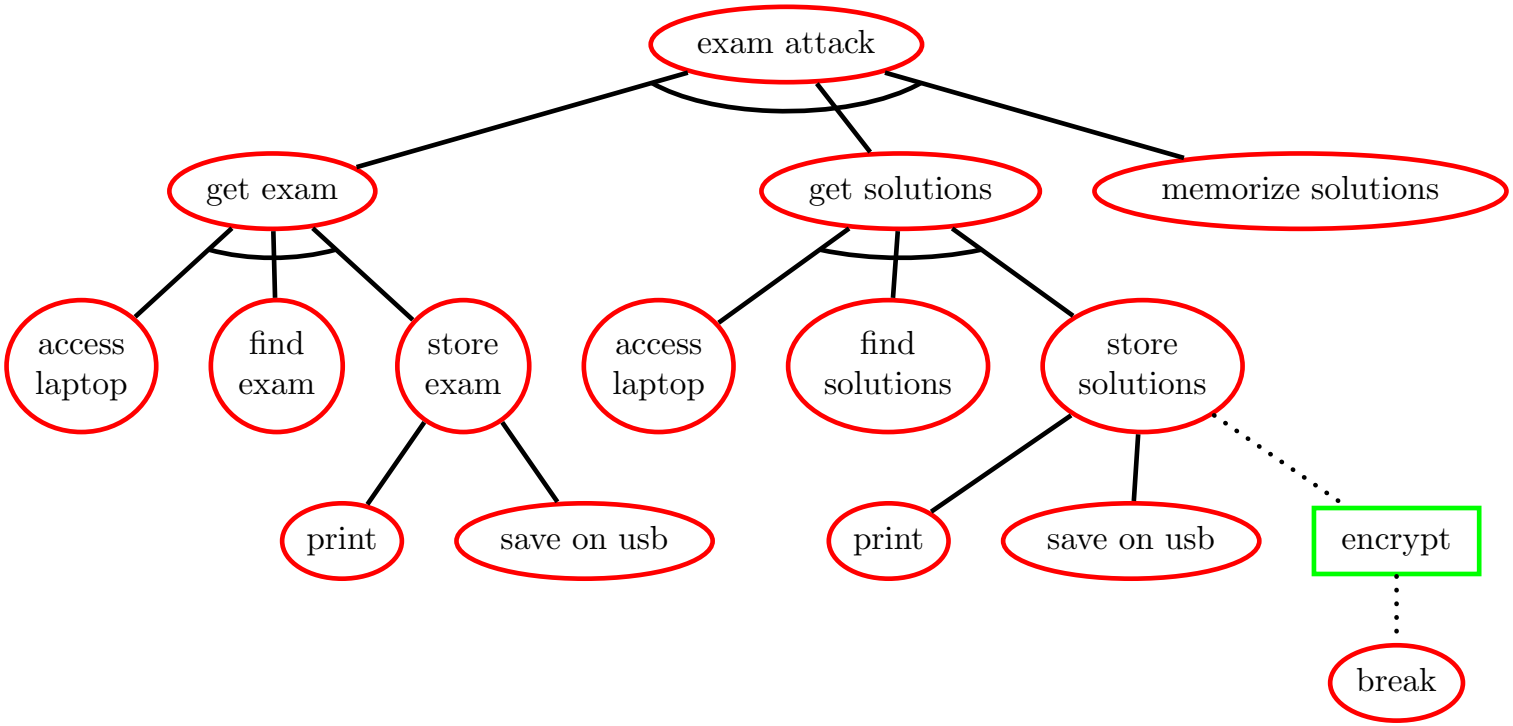
p – proponent – the root actor
o – opponent – the other actor
$\mathbb{B}$ – set of basic actions partitioned into $\mathbb{B}^{\mathrm{p}}$ and $\mathbb{B}^{\mathrm{o}}$

ADTrees are terms of the form T^{p} generated by the grammar

$$T^{\mathrm{p}}: \quad \mathtt{b}^{\mathrm{p}} \mid \mathtt{OR}^{\mathrm{p}}(T^{\mathrm{p}}, \ldots, T^{\mathrm{p}}) \mid \mathtt{AND}^{\mathrm{p}}(T^{\mathrm{p}}, \ldots, T^{\mathrm{p}}) \mid \mathtt{C}^{\mathrm{p}}(T^{\mathrm{p}}, T^{\mathrm{o}})$$
$$T^{\mathrm{o}}: \quad \mathtt{b}^{\mathrm{o}} \mid \mathtt{OR}^{\mathrm{o}}(T^{\mathrm{o}}, \ldots, T^{\mathrm{o}}) \mid \mathtt{AND}^{\mathrm{o}}(T^{\mathrm{o}}, \ldots, T^{\mathrm{o}}) \mid \mathtt{C}^{\mathrm{o}}(T^{\mathrm{o}}, T^{\mathrm{p}})$$

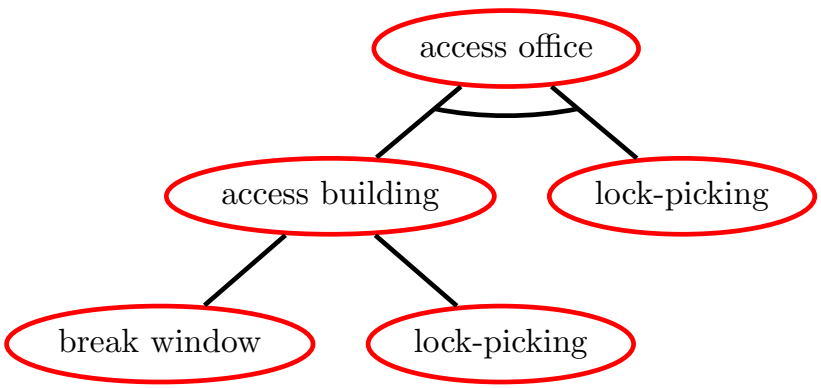
where $\mathtt{b}^{\mathrm{p}} \in \mathbb{B}^{\mathrm{p}}$, $\mathtt{b}^{\mathrm{o}} \in \mathbb{B}^{\mathrm{o}}$

Example



$$\mathtt{AND}^{\mathtt{p}}\Big(\mathtt{AND}^{\mathtt{p}}\Big(\mathtt{lapt},\mathtt{ex},\mathtt{OR}^{\mathtt{p}}(\mathtt{pr},\mathtt{usb})\Big),\ \mathtt{AND}^{\mathtt{p}}\Big(\mathtt{lapt},\mathtt{sol},\mathtt{C}^{\mathtt{p}}(\mathtt{OR}^{\mathtt{p}}(\mathtt{pr},\mathtt{usb}),\mathtt{C}^{\mathtt{o}}(\mathtt{enc},\mathtt{break}))\Big),\ \mathtt{memo}\Big)$$

Existing formalizations of ADTrees

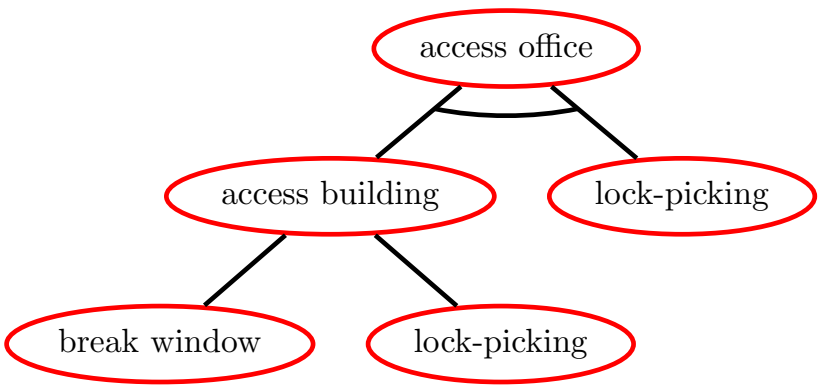


Propositional semantics

Interpreting ADTrees as Boolean formulæ

$(\texttt{window} \vee \texttt{pick}) \wedge \texttt{pick}$

Existing formalizations of ADTrees



Propositional semantics

Interpreting ADTrees as Boolean formulæ

$(\texttt{window} \vee \texttt{pick}) \wedge \texttt{pick}$

Multiset semantics

Interpreting ADTrees as sets of multisets

$\{\{\!|\texttt{window}, \texttt{pick}|\!\}, \{\!|\texttt{pick}, \texttt{pick}|\!\}\}$

Bottom-up algorithm for quantifying attacks

An attribute α is composed of

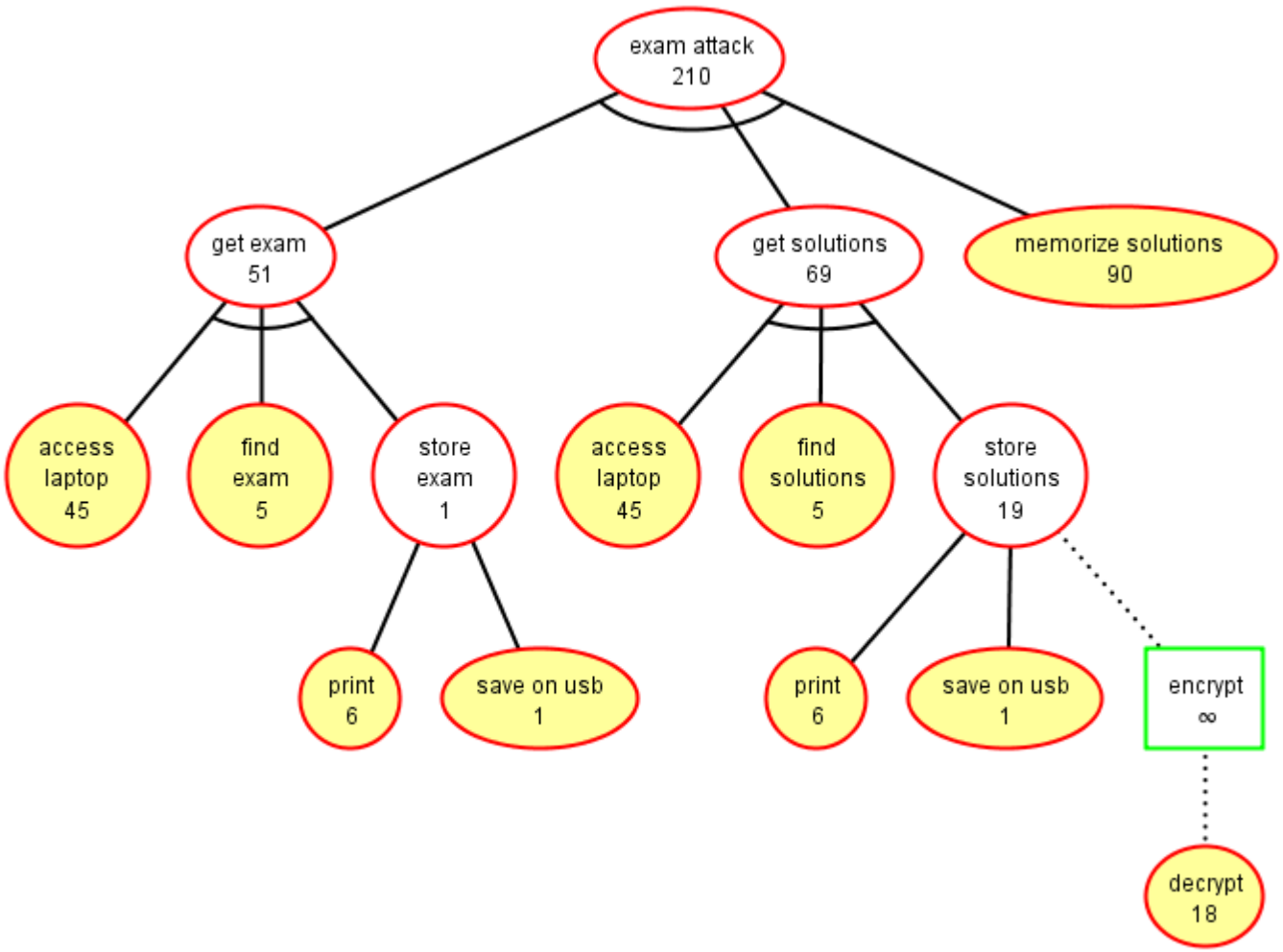
- A set of values D_α
- A basic assignment $\beta_\alpha : \mathbb{B} \to D_\alpha$
- An attribute domain $A_\alpha = (D_\alpha, \mathtt{OR}^{\mathtt{p}}_\alpha, \mathtt{AND}^{\mathtt{p}}_\alpha, \mathtt{OR}^{\mathtt{o}}_\alpha, \mathtt{AND}^{\mathtt{o}}_\alpha, \mathtt{C}^{\mathtt{p}}_\alpha, \mathtt{C}^{\mathtt{o}}_\alpha)$, where $\mathtt{OP}^{\mathtt{s}}_\alpha : D^k_\alpha \to D_\alpha$ is an internal operation on D_α, for $\mathtt{OP} \in \{\mathtt{OR}, \mathtt{AND}, \mathtt{C}\}$

The bottom-up algorithm for α assigns values from D_α to ADTrees

$$\alpha(\mathtt{b}) = \beta_\alpha(\mathtt{b}) \qquad \alpha(\mathtt{OP}^{\mathtt{s}}(T^{\mathtt{s}}_1, \ldots, T^{\mathtt{s}}_k)) = \mathtt{OP}^{\mathtt{s}}_\alpha(\alpha(T^{\mathtt{s}}_1), \ldots, \alpha(T^{\mathtt{s}}_k))$$

Minimal time to attack

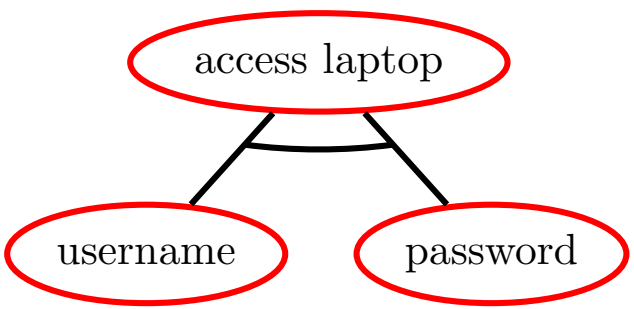
$$A_{\text{time}} = (\mathbb{N} \cup \{+\infty\}, \min, +, +, \min, +, \min)$$



Outline

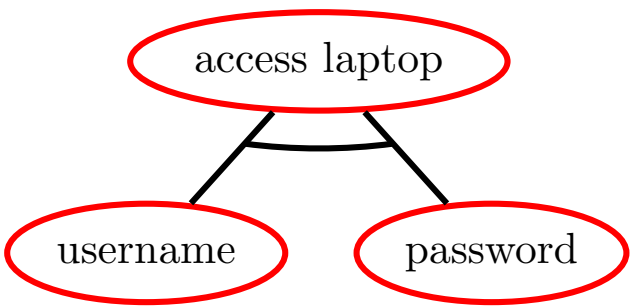
Refinement issue

Incomplete refinement

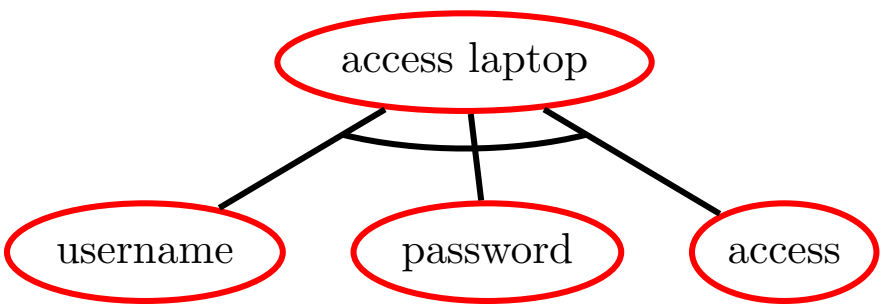


Refinement issue

Incomplete refinement

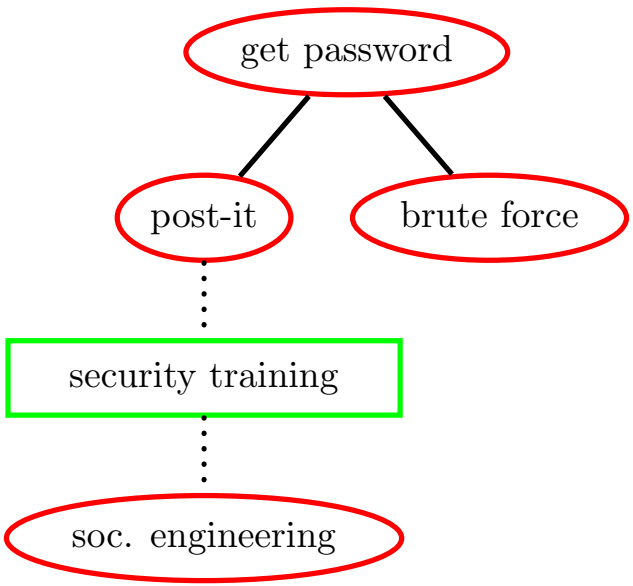


Complete refinement



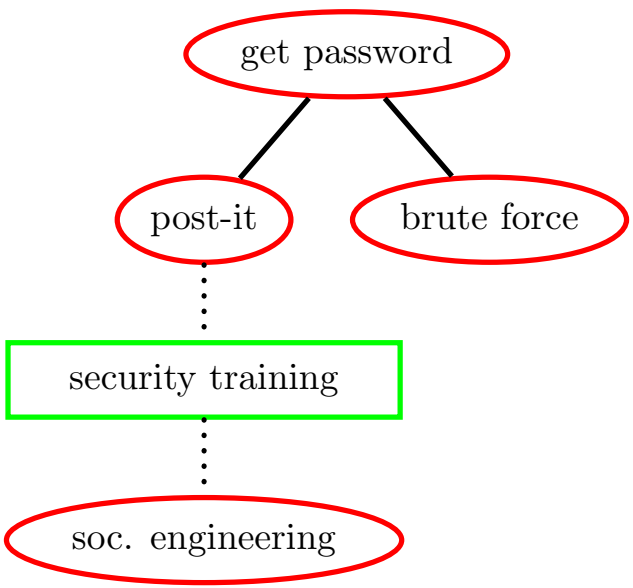
Counter issue

Incorrect countering

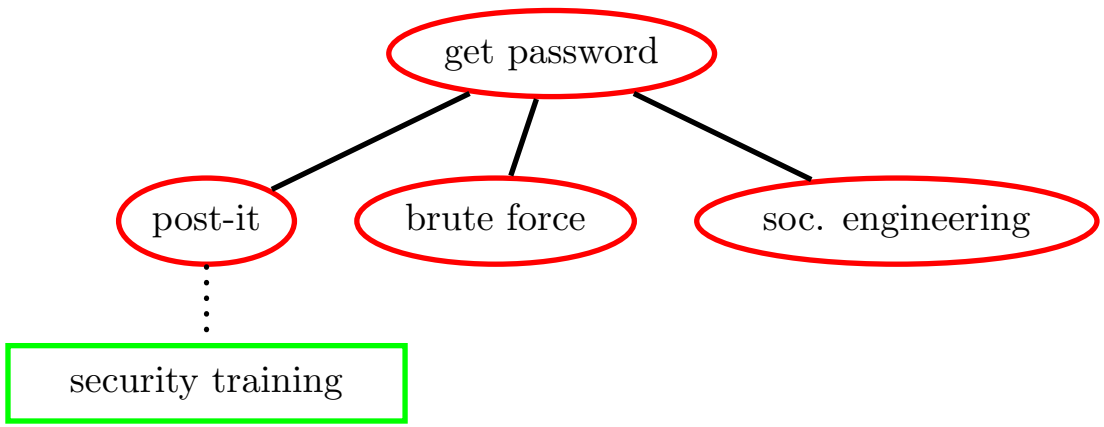


Counter issue

Incorrect countering

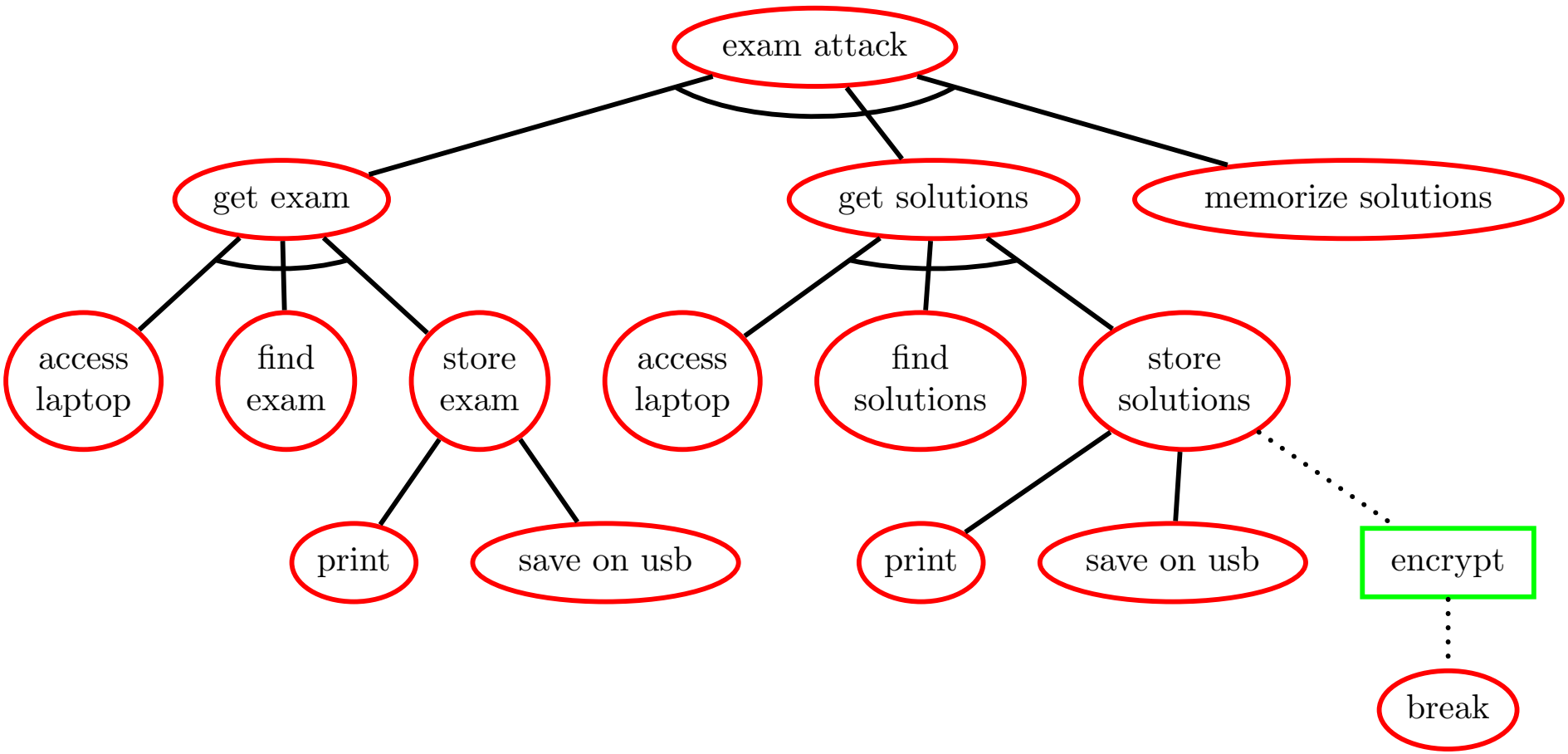


Correct countering



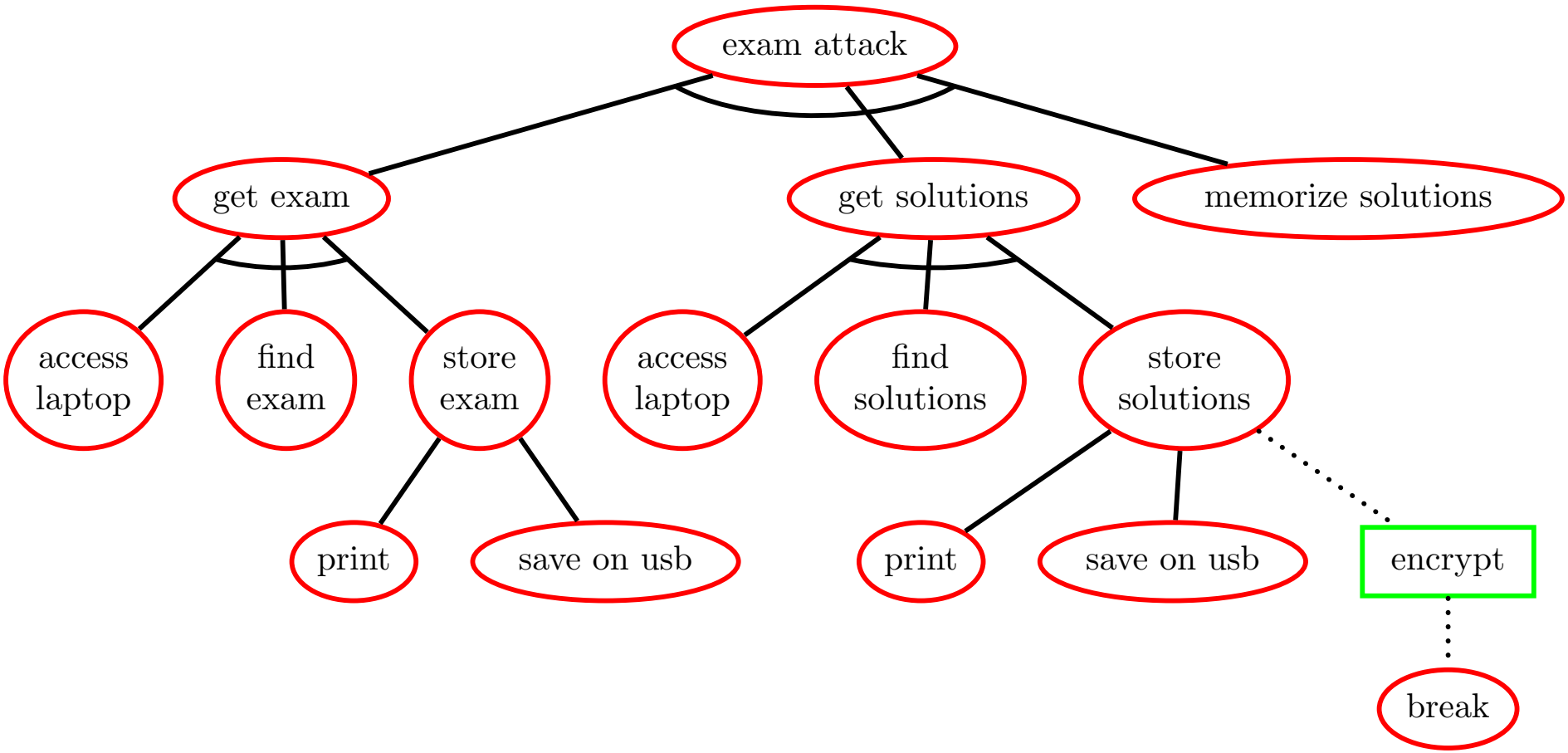
Repeated labels issue

print – time is different for the two print actions



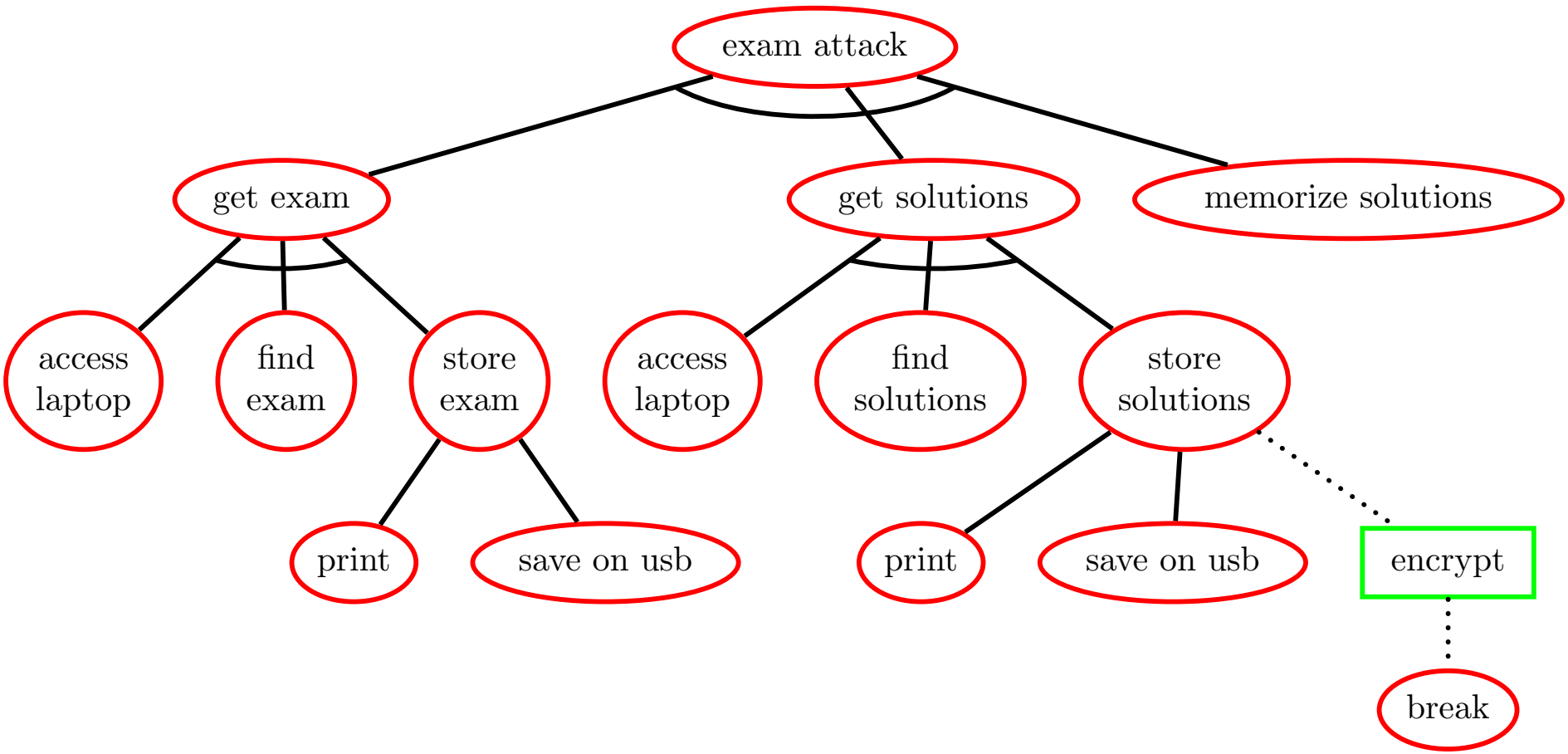
Repeated labels issue

save on usb – one needs to save twice



Repeated labels issue

access laptop – one needs to access the laptop only once



Outline

Clones and twins

Cloned nodes

For two nodes with the same label, when activating one means activating the other one, we say that the two nodes are cloned.

- Cloned nodes represent exactly the same instance of an action
- Deactivating one of the cloned nodes deactivates all of them

Clones and twins

Cloned nodes

For two nodes with the same label, when activating one means activating the other one, we say that the two nodes are cloned.

- Cloned nodes represent exactly the same instance of an action
- Deactivating one of the cloned nodes deactivates all of them

Example: `access laptop` node

Clones and twins

Cloned nodes

For two nodes with the same label, when activating one means activating the other one, we say that the two nodes are cloned.

- Cloned nodes represent exactly the same instance of an action
- Deactivating one of the cloned nodes deactivates all of them

Example: `access laptop` node

Twin nodes

For two nodes with the same label, when activating one does not activate the other one, we say that the two nodes are twins.

- Each individual twin node represents a separated instance of an action
- All twin nodes need to be deactivated separately

Clones and twins

Cloned nodes

For two nodes with the same label, when activating one means activating the other one, we say that the two nodes are cloned.

- Cloned nodes represent exactly the same instance of an action
- Deactivating one of the cloned nodes deactivates all of them

Example: `access laptop` node

Twin nodes

For two nodes with the same label, when activating one does not activate the other one, we say that the two nodes are twins.

- Each individual twin node represents a separated instance of an action
- All twin nodes need to be deactivated separately

Example: `save on usb` node

Motivation

Our goal is to define well-formed ADTrees in a way to

- Allow for the re-use of libraries of trees
- Enable merging of several trees
- Prohibit peculiar, non-intuitive labels resulting from relabeling
- Keep the attribute basic assignment as concise as possible
- Distinguish clones from twins

Extended labeling

Labels as pairs: goal + index

Labels are pairs in $\mathbb{G} \times \Gamma$, where

- $\mathbb{G}$ is a typed set of goals containing $\mathbb{B}$
- Γ is a finite set of indices

Old label g becomes a pair (g, γ)

- $\mathrm{g} \in \mathbb{G}$ describes the goal to be achieved
- $\gamma \in \Gamma$ is an index allowing us to differentiate clones from twins

Extended labeling

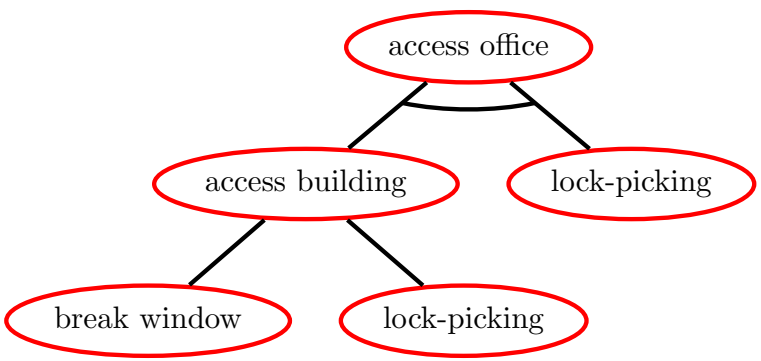
Labels as pairs: goal + index

Labels are pairs in $\mathbb{G} \times \Gamma$, where

- $\mathbb{G}$ is a typed set of goals containing $\mathbb{B}$
- Γ is a finite set of indices

Old label g becomes a pair (g, γ)

- $g \in \mathbb{G}$ describes the goal to be achieved
- $\gamma \in \Gamma$ is an index allowing us to differentiate clones from twins



Extended labeling

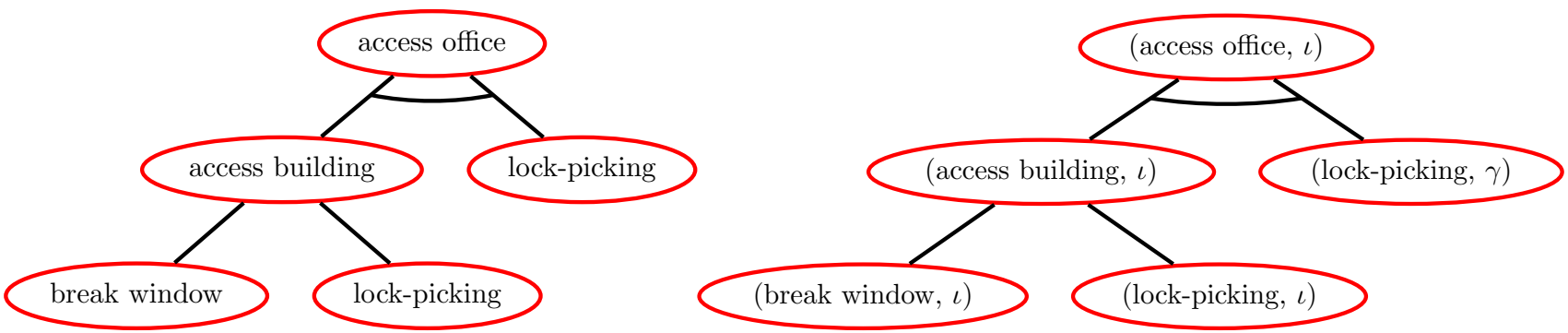
Labels as pairs: goal + index

Labels are pairs in $\mathbb{G} \times \Gamma$, where

- $\mathbb{G}$ is a typed set of goals containing $\mathbb{B}$
- Γ is a finite set of indices

Old label g becomes a pair (g, γ)

- $g \in \mathbb{G}$ describes the goal to be achieved
- $\gamma \in \Gamma$ is an index allowing us to differentiate clones from twins



Grammar for well-formed ADTrees

Well-formed ADTrees are generated by the grammar

$$
\begin{aligned}
T^{\mathtt{p}} : (\mathtt{b}^{\mathtt{p}}, \gamma) \;&|\; \mathtt{OR}^{\mathtt{p}}[(\mathtt{g}, \gamma)](T^{\mathtt{p}}, \ldots, T^{\mathtt{p}}) \\
&|\; \mathtt{AND}^{\mathtt{p}}[(\mathtt{g}, \gamma)](T^{\mathtt{p}}, \ldots, T^{\mathtt{p}}) \\
&|\; \mathtt{C}^{\mathtt{p}}(T^{\mathtt{p}}, T^{\mathtt{o}}) \\
\\
T^{\mathtt{o}} : (\mathtt{b}^{\mathtt{o}}, \gamma) \;&|\; \mathtt{OR}^{\mathtt{o}}[(\mathtt{g}, \gamma)](T^{\mathtt{o}}, \ldots, T^{\mathtt{o}}) \\
&|\; \mathtt{AND}^{\mathtt{o}}[(\mathtt{g}, \gamma)](T^{\mathtt{o}}, \ldots, T^{\mathtt{o}}) \\
&|\; \mathtt{C}^{\mathtt{o}}(T^{\mathtt{o}}, T^{\mathtt{p}})
\end{aligned}
$$

where $\mathtt{b}^{\mathtt{p}} \in \mathbb{B}^{\mathtt{p}}$, $\mathtt{b}^{\mathtt{o}} \in \mathbb{B}^{\mathtt{o}}$

Well-formed ADTree

An ADTree is well-formed iff the following conditions are satisfied

T is of the proponent's type

$$
\begin{aligned}
T^{\mathtt{p}} : (\mathtt{b}^{\mathtt{p}}, \gamma) \quad &| \quad \mathtt{OR}^{\mathtt{p}}[(\mathtt{g}, \gamma)](T^{\mathtt{p}}, \ldots, T^{\mathtt{p}}) \\
&| \quad \mathtt{AND}^{\mathtt{p}}[(\mathtt{g}, \gamma)](T^{\mathtt{p}}, \ldots, T^{\mathtt{p}}) \\
&| \quad \mathtt{C}^{\mathtt{p}}(T^{\mathtt{p}}, T^{\mathtt{o}})
\end{aligned}
$$

Well-formed ADTree

An ADTree is well-formed iff the following conditions are satisfied

All refinements are correct and complete

Let g_i be the goal of the root node of T_i

- $\mathtt{OR}^s[(g, \gamma)](T_1^s, \ldots, T_k^s)$
  Goal g is achieved iff at least one of the subgoals g_i is achieved
- $\mathtt{AND}^s[(g, \gamma)](T_1^s, \ldots, T_k^s)$
  Goal g is achieved iff all of the subgoals g_i are achieved

Well-formed ADTree

An ADTree is well-formed iff the following conditions are satisfied

Countering subtree disables the goal of the node it is attached to

Let g_i be the goal of the root node of T_i

- $C^s(T_1^s, T_2^{\bar{s}})$

  If g_2 is achieved then g_1 cannot be achieved

Well-formed ADTree

An ADTree is well-formed iff the following conditions are satisfied

All counters are correctly placed

Let g_i be the goal of the root node of T_i

- $C^{s}(T_1^{s}, C^{\bar{s}}(T_2^{\bar{s}}, T_3^{s}))$

  Achieving g_3 does not achieve any goal of type s from T_1,
  in particular, achieving g_3 does not replace achieving g_1

Well-formed ADTree

An ADTree is well-formed iff the following conditions are satisfied

Cloned nodes represent the same events

- Trees rooted in cloned nodes are equivalent wrt the used semantics
- Trees rooted in cloned nodes yield the same attribute value

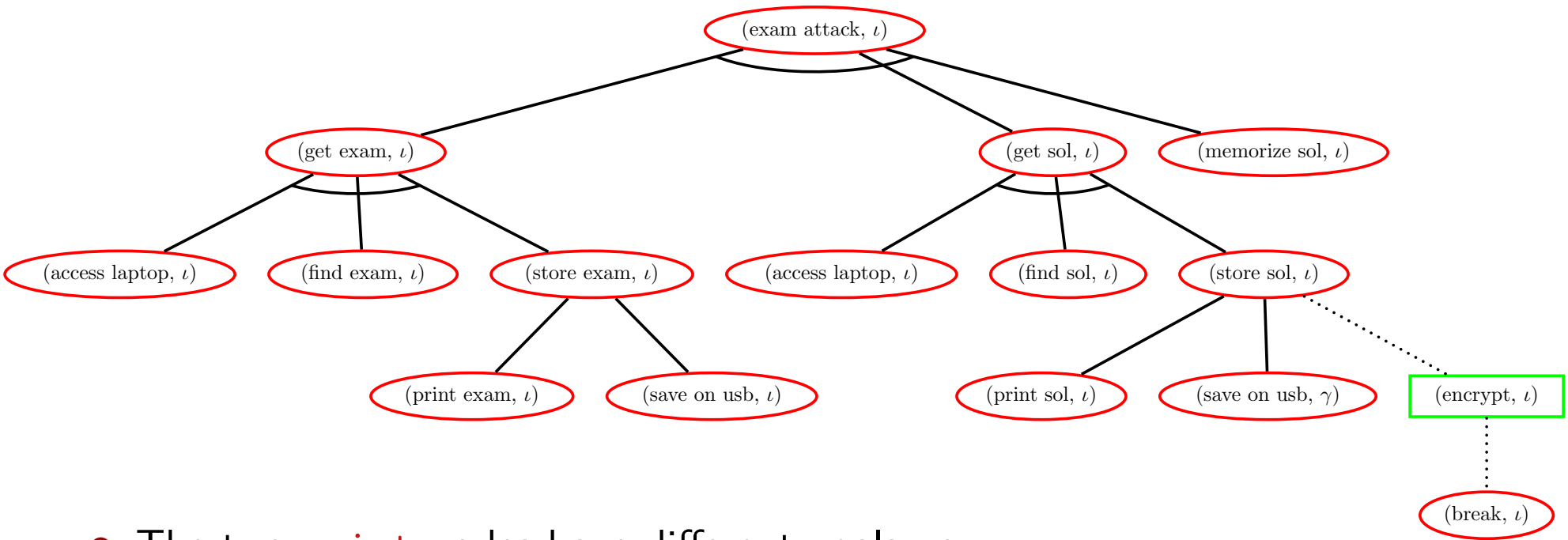
Well-formed ADTree

An ADTree is well-formed iff the following conditions are satisfied

Twin nodes represent similar events

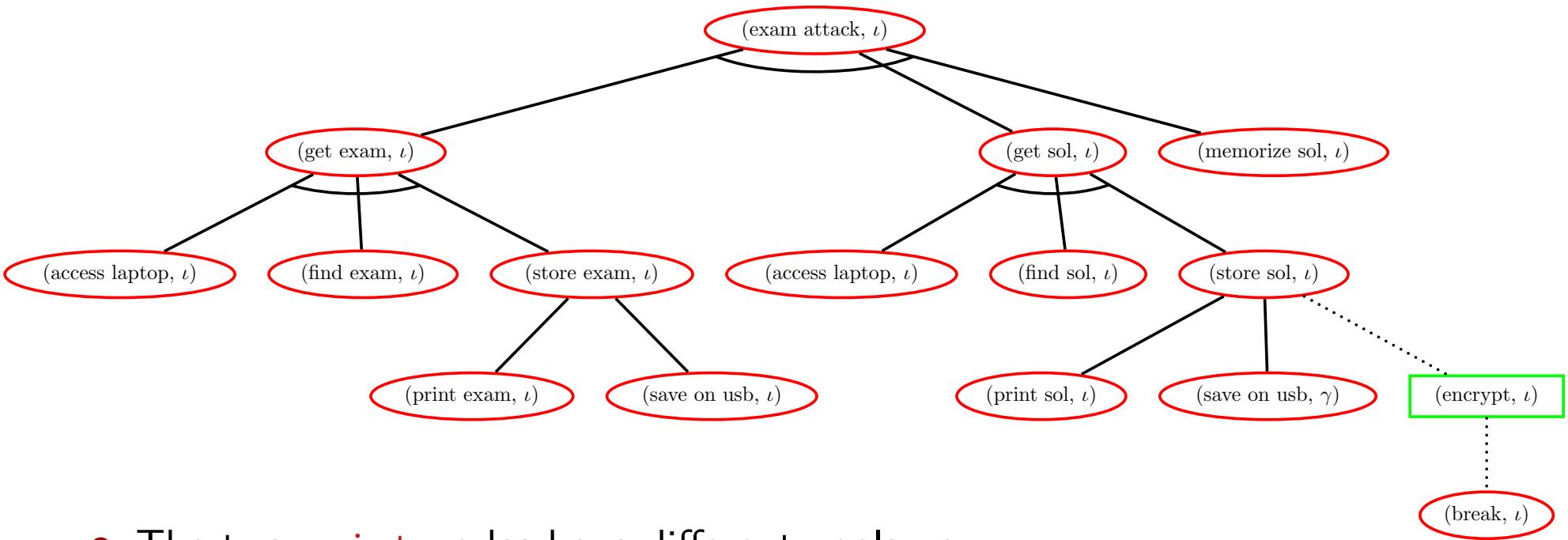
- Trees rooted in twin nodes have equivalent refining subtrees
- The refining subtrees of trees rooted in twin nodes yield the same attribute value

Well-formed example



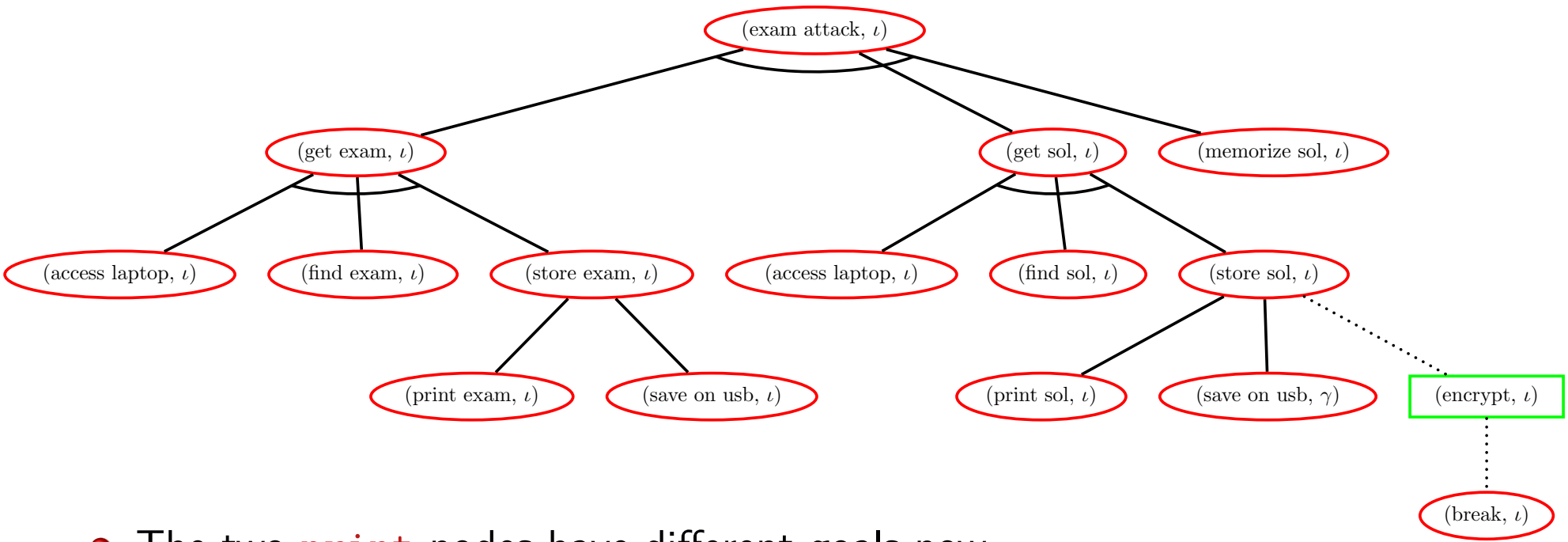
- The two `print` nodes have different goals now

Well-formed example



- The two `print` nodes have different goals now
- The two `access laptop` nodes are clones (the same indices)

Well-formed example

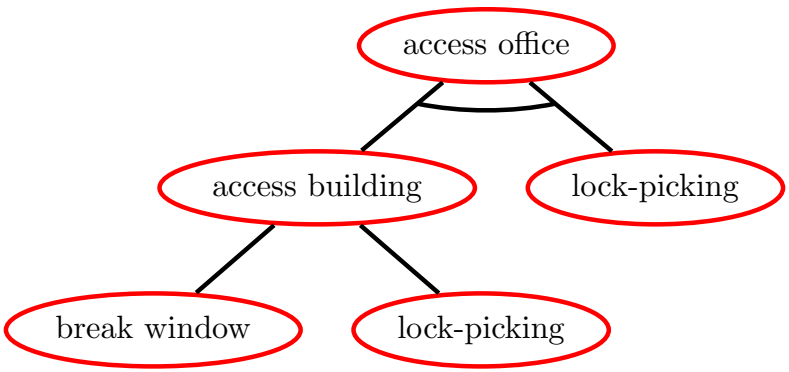


- The two `print` nodes have different goals now
- The two `access laptop` nodes are clones (the same indices)
- The two `save on usb` nodes are twins (different indices)

Set semantics for well-formed ADTrees

Replace the multisets by the sets of pairs (goal, index)

Classical multiset semantics

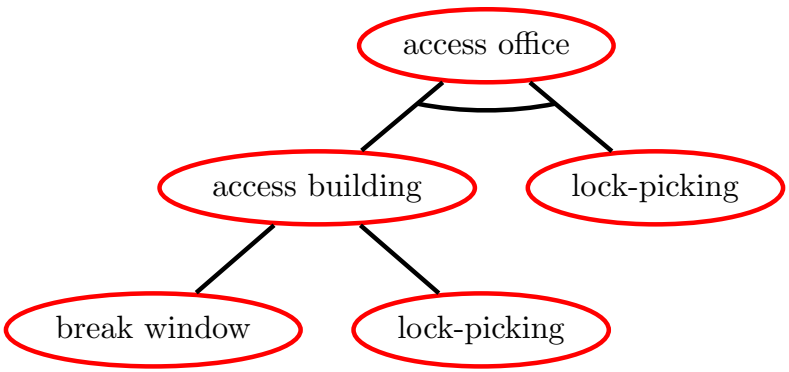


$$\Big\{ \{\!|\texttt{window}, \texttt{pick}|\!\}, \{\!|\texttt{pick}, \texttt{pick}|\!\} \Big\}$$

Set semantics for well-formed ADTrees

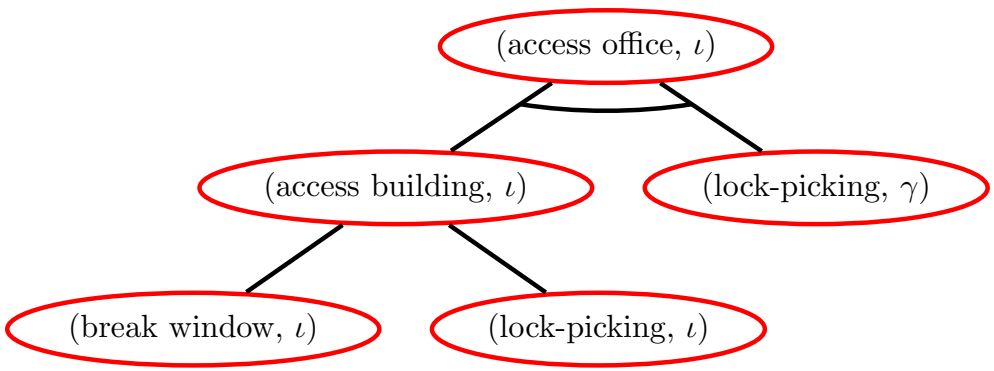
Replace the multisets by the sets of pairs (goal, index)

Classical multiset semantics



$$\Big\{ \{\!|\texttt{window}, \texttt{pick}|\!\}, \{\!|\texttt{pick}, \texttt{pick}|\!\} \Big\}$$

Set semantics for well-formed ADTrees



$$\Big\{ \{(\texttt{window}, \iota), (\texttt{pick}, \gamma)\}, \{(\texttt{pick}, \iota), (\texttt{pick}, \gamma)\} \Big\}$$

Quantification of well-formed ADTrees

Quantification on the set semantics

Let $A_\alpha = (D_\alpha, \mathtt{OR}^{\mathtt{p}}_\alpha, \mathtt{AND}^{\mathtt{p}}_\alpha, \mathtt{OR}^{\mathtt{o}}_\alpha, \mathtt{AND}^{\mathtt{o}}_\alpha, \mathtt{C}^{\mathtt{p}}_\alpha, \mathtt{C}^{\mathtt{o}}_\alpha)$ be an attribute domain

- Basic assignment
  - Assign values to basic goals
  - Cloned and twin nodes get the same value
- Compute the set semantics
  - $\mathcal{S}(T) = \bigcup_{i=1}^{l} \left\{ \left(\bigcup_{j=1}^{n_i} \{(\mathtt{p}_{ij}, \gamma_{ij})\}, \bigcup_{j=1}^{m_i} \{(\mathtt{o}_{ij}, \gamma_{ij})\} \right) \right\}$
- Compute the value for the entire tree
  - $\alpha(T) = (\mathtt{OR}^{\mathtt{p}}_\alpha)_{i=1}^{l} \left(\mathtt{C}^{\mathtt{p}}_\alpha \left((\mathtt{AND}^{\mathtt{p}}_\alpha)_{j=1}^{n_i} \beta_\alpha(\mathtt{p}_{ij}), (\mathtt{OR}^{\mathtt{o}}_\alpha)_{j=1}^{m_i} \beta_\alpha(\mathtt{o}_{ij}) \right) \right)$

Outline

Wrap-up

Problems

- Lack of guidelines for the modeler
- Simplistic formalizations of ADTrees
- Repeated labels

Wrap-up

Problems

- Lack of guidelines for the modeler
- Simplistic formalizations of ADTrees
- Repeated labels

Objectives

- Re-usability of libraries
- Intuitive labels
- Efficient quantification

Wrap-up

Problems

- Lack of guidelines for the modeler
- Simplistic formalizations of ADTrees
- Repeated labels

Objectives

- Re-usability of libraries
- Intuitive labels
- Efficient quantification

Solutions

- Extended grammar for ADTrees
- Definition of cloned and twin nodes
- Formalization of well-formed ADTrees

Problems still open

Tool support

- Automated creation of ADTrees from a system description
- Implementation of well-formedness checker

Problems still open

Tool support

- Automated creation of ADTrees from a system description
- Implementation of well-formedness checker

Expressive power

- Preventive and reactive countermeasures
- Dependencies between the nodes

Thank you for your attention

Thank you for your attention

Main credits for this work go to Angèle!