A Formal Foundation for XrML

What’s XrML?
- An XML-based language for writing software licenses. Language specification includes:
  - syntax,
  - English description of components, and
  - algorithm for determining if a permission follows from a set of licenses.
- International standard committees (e.g. MPEG, OASIS) are modifying XrML to build application-specific languages that are designed for use across entire industries.
- Microsoft, OverDrive, and DMDsecure plan to build XrML compliant products (Adobe Systems, HP labs, Xerox, Barnesandnobel.com, and Time Warner Trade Publishing have also voiced their support).

Bottom Line: The people who write/enforce software licenses in industry say XrML meets their needs.

Problem: No Formal Semantics = Bugs

According to the XrML Algorithm:
- The two statements:
  - Alice may issue g and
  - if Alice may issue g, then Bob may issue g
do not imply that Bob may issue g.
- The two statements:
  - Alice may issue g
  - if Alice and Bob together may issue g, then Charlie may issue гimplies that Alice and Bob together may issue g.
But, the two statements do not imply that Charlie may issue g. Finding these bugs required a close examination of the algorithm.

Bottom Line: XrML’s algorithm is buggy. If we had formal semantics, we could prove that a revised algorithm was correct.

Our Approach: Translate XrML licenses into formulas in a logic (that has formal semantics).

Which Logic?
- We use first-order logic where variables range over terms.
- XrML statements are of the form ‘if $\varphi$ is valid, then $\psi$ is valid’
  - E.g., if ‘Alice may issue g’ is a logical consequence of L, then ‘Bob may issue g’ is a logical consequence of L where L is a set of licenses.
  - We have a validity modality such that $\text{Val}(\varphi)$ is true in a model m iff $\varphi$ is true in all models m'.

Bottom Line: We translate XrML licenses to formulas in a variant of first-order logic with a validity modality.

Using First-order Logic to Reason about Policies

Encoding Policies
- A policy says what is or what is not permitted.
- Examples policies include:
  - ‘All information on this site may be copied.’
  - ‘The tickets may not be refunded.’
- A policy has the form:
  - $\forall x_1, \ldots, x_m (f \Rightarrow \neg l)$

Encoding the Environment
- An environment is a conjunction of ground literals.
- An environment is a conjunction of universal formulas.

Encoding Queries
- A query asks if a policy is permitted.
- A query asks if an environment is allowed.

Complexity/Tractability
- Determining if a permission follows from a set of XrML licenses is an NP-hard problem.
- We need to find a tractable fragment that is expressively real for applications.
- To find a tractable fragment, we apply our results from the spring.

Partially supported by the NSF grant "Software Quality and Infrastructure Protection for Diffuse Computing" through ONR grant N00014-01-1-0795.

Researchers: Joseph Y. Halpern and Vicky Weissman; Computer Science Dept., Cornell University