

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 g
 imply 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 ϕ is valid, then ψ 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.
- \Rightarrow We have a validity modality such that **Val(ϕ)** is true in a model m iff ϕ 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.
Partially supported by the CIP/SW URI

"Software Quality and Infrastructure Protection for Diffuse Computing" through ONR grant N00014-01-1-0795.

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 sufficiently expressive for real applications.
- To find a tractable fragment, we apply our results from the spring.

At our last review...

Using First-order Logic to Reason about Policies

Background: Policies say what is and what is not permitted.

- Sample policies include:
- 'All information on this site may be copied.'
 - 'The tickets may not be refunded.'

Goals: To create a logic that

1. can easily capture the policies that many people want to discuss
2. can efficiently determine what is allowed and what is forbidden
3. is accessible to non-logicians

Why bother?: We want to promote the dissemination of ideas, while still respecting intellectual property rights. To do this, we must be able to state what should be shared (i.e. what's permitted) and what constitutes a violation of a person's rights (i.e. what's not permitted).

Our Approach:

Encoding Policies

A policy says what is (or what is not) permitted.

- A policy has the form:
 $\forall x_1, \dots, \forall x_n (f \Rightarrow (\neg) \text{Permitted}(t_{ag}, t_{ac}))$
 where
- f is a conjunction of literals;
 - t_{ag} is an agent, t_{ac} is an action, both are terms;
 - **Permitted**(t_{ag}, t_{ac}) means t_{ag} may do t_{ac}

Encoding the Environment

The environment (env) gives basic facts about the world.

- An environment is a conjunction of
 - ground literals
e.g. **Student**(Alice)
 - universal formulas;
e.g. $\forall x (\text{Man}(x) \Rightarrow \text{Woman}(x))$

Encoding Queries

Assume an environment E and a policy set $P = \{p_1, \dots, p_n\}$, is c_1 allowed/forbidden to do c_2 ?

Is $E \wedge p_1 \wedge \dots \wedge p_n \Rightarrow (\neg) \text{Permitted}(c_1, c_2)$ a valid formula?

Key Idea: Bipolarity

- 2 literals l and l' are unifiable if $\exists \sigma. l\sigma = l'\sigma$.
- A literal l is bipolar in a formula f (in CNF) if l is in f and there is a literal l' in f such that l and -l' are unifiable, (assume no shared variables).

Complexity

If

- the env. E has only ground literals,
- for the policy set $P = \{p_1, \dots, p_n\}$ there are no bipolars in $p_1 \wedge \dots \wedge p_n$,
- no variable is only on a policy's lhs,

Then our queries take $|P||E|$ time to ans.

Relaxing Restrictions

- If the variable restriction isn't met, then problems are NP in the number of variables in any one policy.
- Under reasonable assumptions, answering queries takes quadratic time, even if the env. has universal formulas.