



A logic for reasoning about digital rights

Riccardo Pucella, Vicky Weissman
{riccardo,vickyw}@cs.cornell.edu
Cornell University



Licenses

A *license* states **the exact conditions** under which **a resource may be used**.

- **Examples:**

- The client must **sign a waiver** before **downloading beta-version software**.
- The client must **sign a lease** and either **pay \$490 on the first day or \$500 on the second day of each month** to **live in an apartment**.



Reasoning about licenses- Properties

Does a property hold for a given set of licenses, regardless of the client's actions?

- Examples of properties include:
 - `A religious work may never be viewed after sunset.'
 - `If a client uses a resource, then the client is obligated to pay for the use at some time.'
- Depending on the licenses, a property may or may not be easy to check.



Reasoning about licenses- Specifications

Does a property hold for a given set of licenses and client behavior?

- Examples of specifications include:
 - The client never uses a resource illegally.
 - The client is never obligated to pay interest on any debts.
- Specifications may or may not be easy to check, depending on the given info.



Our goal

To design a logic that we can use to:

1. easily state interesting properties and specifications;
2. prove that a property holds (or a specification is met) for a given license set (and client behavior).



Logic features

The logic needs to talk about:

- licenses,
- client behavior wrt a license,
- time – temporal operators,
- permission and obligation.



Licenses

- Follow lead from Gunter, Weeks, Wright 'Models and languages for digital rights', 2001.
 - Licenses are sets of traces.
 - Each trace describes an action sequence that the client could do to fulfill the license.
- Can write licenses in various languages. We'll use regular expressions.

$$l ::= a \mid l_1 l_2 \mid l^* \mid l_1 \cup l_2$$

where a is an action.



Licenses

- Follow lead from Gunter, Weeks, Wright 'Models and languages for digital rights', 2001.
 - Licenses are **sets of traces**.
 - Each trace describes an action sequence that the client could do to fulfill the license.
- Can write licenses in various languages. We'll use regular expressions.

$$l ::= a \mid l_1 l_2 \mid l^* \mid l_1 \cup l_2$$

where **a** is an action.



Licenses

- Follow lead from Gunter, Weeks, Wright 'Models and languages for digital rights', 2001.
 - Licenses are sets of traces.
 - Each trace describes an **action sequence** that the client could do to fulfill the license.
- Can write licenses in various languages. We'll use regular expressions.

$$I ::= \mathbf{a} \mid I_1 I_2 \mid I^* \mid I_1 \cup I_2$$

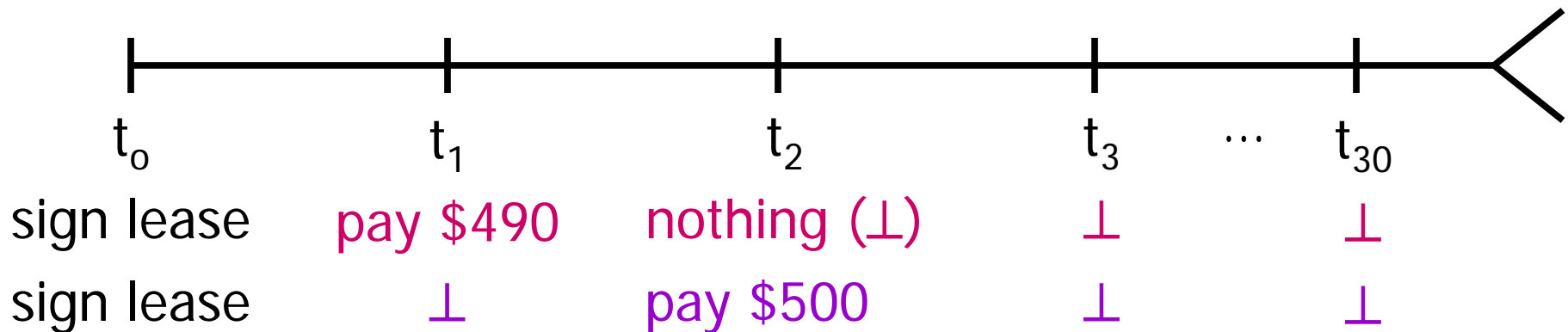
where \mathbf{a} is an action.

Lease Example

Recall the lease example:

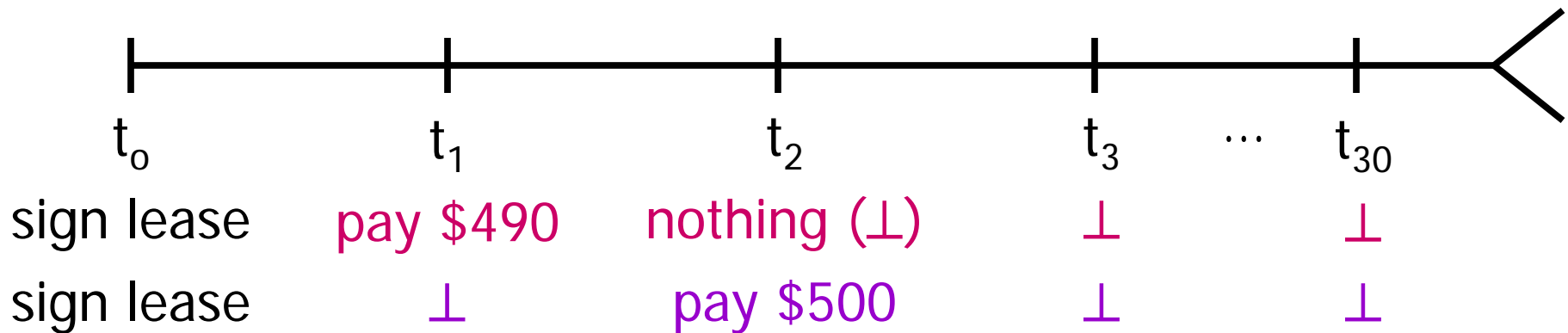
The client must sign a lease and either pay \$490 on the first day or \$500 on the second day of each month to live in an apartment.

Viewed as a set of traces:



Encoding action sequences

Traces:



License: $I_1 = \text{pay } \$490 \perp \dots \perp$

$I_2 = \perp \text{ pay } \$500 \perp \dots \perp$

$I = \text{sign lease } (I_1 \cup I_2)^*$



What a client can do

Client can do any action α of the form:

$\alpha ::=$

(a, n) | do action a wrt license named n

(\bar{a}, n) | do not do action a wrt license n

E.g. The client does not pay \$490 for the lease is written: $(\overline{\text{pay } \$490}, \text{lease})$.



Writing properties and specs.

A formula f has the form:

$f ::=$

α | client does action expression α

$P(\alpha)$ | client permitted to do α

$n:l$ | license l with name n is issued

$\bigcirc f$ | f | $f_1 \mathbf{U} f_2$ next time, always, until

$f_1 \wedge f_2$ | $\neg f$



Obligation

Can capture obligation using permission.

- A client is obligated to do (a, n) , if she isn't permitted to do any other action, including the do-nothing action \perp , wrt license n .
- The client is obligated to do (a, n) if $P(a, n) \wedge \neg P(\bar{a}, n)$ holds.
- In our logic, client is always permitted to do something, possibly \perp , wrt each license. So, $P(a, n) \wedge \neg P(\bar{a}, n) = \neg P(\bar{a}, n)$



Obligation

Can capture obligation using permission.

- A client is obligated to do (a, n) , if she isn't permitted to do any other action, including the do-nothing action \perp , wrt license n .
- The **client is obligated to do (a, n)** if $P(a, n) \wedge \neg P(\bar{a}, n)$ holds.
- In our logic, client is always permitted to do something, possibly \perp , wrt each license. So, $P(a, n) \wedge \neg P(\bar{a}, n) = \neg P(\bar{a}, n)$



Obligation

Can capture obligation using permission.

- A client is obligated to do (a, n) , if she isn't permitted to do any other action, including the do-nothing action \perp , wrt license n .
- The client is obligated to do (a, n) if $P(a, n) \wedge \neg P(\bar{a}, n)$ holds.
- In our logic, client is always permitted to do something, possibly \perp , wrt each license. So, $P(a, n) \wedge \neg P(\bar{a}, n) = \neg P(\bar{a}, n)$



Obligation

Can capture obligation using permission.

- A client is obligated to do (a, n) , if she isn't permitted to do any other action, including the do-nothing action \perp , wrt license n .
- The client is obligated to do (a, n) if $P(a, n) \wedge \neg P(\bar{a}, n)$ holds.
- In our logic, client is always permitted to do something, possibly \perp , wrt each license. So, $P(a, n) \wedge \neg P(\bar{a}, n) = \neg P(\bar{a}, n)$

Examples

- Property: When the lease is issued, the client must sign it.

$$\text{lease}:I \Rightarrow \neg P(\overline{\text{sign}}, \text{lease})$$

- Spec.: The client doesn't violate the lease. This means that once the lease is issued, the client meets all obligations and only does what's permitted. For all actions a:

$$\text{lease}:I \Rightarrow [(\neg P(\overline{a}, \text{lease})) \Rightarrow (a, \text{lease})] \wedge ((a, \text{lease}) \Rightarrow P((a, \text{lease})))$$

Examples

- Property: **When the lease is issued**, the client must sign it.

$$\text{lease:I} \Rightarrow \neg P(\overline{\text{sign}}, \text{lease})$$

- Spec.: The client doesn't violate the lease. This means that once the lease is issued, the client meets all obligations and only does what's permitted. For all actions a :
$$\text{lease:I} \Rightarrow [(\neg P(\overline{a}, \text{lease})) \Rightarrow (a, \text{lease})] \wedge ((a, \text{lease}) \Rightarrow P((a, \text{lease})))$$

Examples

- Property: When the lease is issued, **the client must sign it.**

$$\text{lease}:I \Rightarrow \neg P(\overline{\text{sign}}, \text{lease})$$

- Spec.: The client doesn't violate the lease. This means that once the lease is issued, the client meets all obligations and only does what's permitted. For all actions a :

$$\text{lease}:I \Rightarrow [(\neg P(\overline{a}, \text{lease})) \Rightarrow (a, \text{lease})] \wedge ((a, \text{lease}) \Rightarrow P((a, \text{lease})))$$

Examples

- Property: When the lease is issued, the client must sign it.

$$\text{lease}:I \Rightarrow \neg P(\overline{\text{sign}}, \text{lease})$$

- Spec.: The client doesn't violate the lease. This means that **once the lease is issued**, the client meets all obligations and only does what's permitted. For all actions a:

$$\text{lease}:I \Rightarrow [(\neg P(\overline{a}, \text{lease})) \Rightarrow (a, \text{lease})] \wedge ((a, \text{lease}) \Rightarrow P((a, \text{lease})))$$

Examples

- Property: When the lease is issued, the client must sign it.

$$\text{lease}:I \Rightarrow \neg P(\overline{\text{sign}}, \text{lease})$$

- Spec.: The client doesn't violate the lease. This means that once the lease is issued, **the client meets all obligations** and only does what's permitted. For all actions a :

$$\text{lease}:I \Rightarrow [(\neg P(\bar{a}, \text{lease})) \Rightarrow (a, \text{lease})] \wedge ((a, \text{lease}) \Rightarrow P((a, \text{lease})))$$

Examples

- Property: When the lease is issued, the client must sign it.

$$\text{lease}:I \Rightarrow \neg P(\overline{\text{sign}}, \text{lease})$$

- Spec.: The client doesn't violate the lease. This means that once the lease is issued, the client meets all obligations and **only does what's permitted**. For all actions a:

$$\text{lease}:I \Rightarrow [(\neg P(\overline{a}, \text{lease})) \Rightarrow (a, \text{lease})] \wedge ((a, \text{lease}) \Rightarrow P((a, \text{lease})))$$



Semantics

- Idea from Halpern and Meyden 'A logic for SDSI's linked local name spaces', 2001.
- We have:
 - a run r that says what happens at any time t . Specifically, $r(t) = (L, A)$ where L are the licenses issued and A are the client's actions done at time t .
 - a permission interpretation $P_r(t)$ says what's allowed at time t , based on the run r .



Example

Suppose that at time t in run r

- the following lease is issued:

$I = \text{sign} (\text{pay } \$490 \perp \dots \perp U \perp \text{pay } \$500 \perp \dots \perp)^*$,

- the client signs the lease, and
- the client ignores the credit card.

In this case

$r(t) = ((\{lease:I\}, \{(sign, lease), (\perp, cc)\}))$

and $P_r(t) = \{(sign, lease)\dots\}$.



Truth conditions

Let $r(t) = (L, A)$

$r, t \models (a, n)$ iff $(a, n) \in A$

$r, t \models (\bar{a}, n)$ iff $\exists b \neq a$ such that $(b, n) \in A$

$r, t \models P((a, n))$ iff $(a, n) \in P_r(t)$

$r, t \models P(\bar{a}, n)$ iff $\exists b \neq a$ such that $(b, n) \in P_r(t)$

$r, t \models n:l$ iff $(n:l) \in L$

Of , $f, f_1 \mathbf{U} f_2, f_1 \wedge f_2, \neg f$ have standard meanings.



How hard is reasoning?

We reduce satisfiability for our logic to that for Linear Temporal Logic (LTL).

- Difference between logics:
 - Our logic has $n:I$, α , and $P(\alpha)$
 - LTL has a set of primitives.
- Easy to encode our 'extras' as primitives. E.g. lease:I becomes the primitive $\text{issued(lease, I)'$.
- Also, easy to convert the runs.

But this isn't quite enough....



Encoding implied facts

- Our logic has implicit notions that must be made explicit in LTL. These include:
 - No name refers to more than one license.
 - All the permissions and obligations implied by an issued license.
- The encoding is given in the paper.



Complexity

Given our translation to LTL, we can use well-known results for LTL to show that:

- Validity checking in our logic is PSPACE-complete.
- Determining if a formula φ holds at a particular time t in a given run r takes polynomial time wrt the size of $r(t)$ and exponential time wrt the size of φ .



Conclusions and future work

- We have introduced a formal framework for reasoning about licenses. Small specifications can be analyzed efficiently.
- Framework can be modified easily to handle different license languages that have trace-based semantics.
- Where do we go from here?
 - Use framework to compare different license languages.
 - Compare our framework to other approaches that talk about permissions.
 - Provide an axiomatization for our logic.