Tutorial on Substructural Logic

Hongseok Yang

MICROS, KAIST

Email: hyang@kaist.ac.kr

August 21 2003

Resource-Sensitive Logic

- Logic provides a language to express a property, and a proof system for reasoning about sentences in the language.
- In computer science, we often need to express resource-sensitive properties and to reason about them.
  - I have enough money to buy both a computer and PS2.
  - This sentence can be split into two so that the first part ends with "sentence" and the second part starts with "can".
  - If I obtain all the access capabilities of user A in addition to what I have already, then I can change any files in this computer.
- Substructural logic allows one to express and reason about such resource-sensitive properties.

Substructural Logic

What is substructural logic?

- Model-theoretically, substructural logic is a logic about resource-sensitive connectives.
- Proof-theoretically, substructural logic is a proof system that does not have certain structural rules.

Researches on Substructural Logic

Substructural logic is currently being studied in many areas of computing.

- Separation Logic
- Typed Assembly Language: Alias Type, Stack Typing
- Logic for Hierarchical Storage
- Logic for Processes: Ambient Logic
- Query Language for Semi-structured Data
- Resource-sensitive Logic Programming

Outline

1. Model of Resources
2. Syntax and Semantics of Formulas
3. Proof Rules and Soundness

1. Model of Resources

About what do we want to reason? What are resources?

Monoid Model of Resources

We model resources using a monoid $\mathcal{M}$.

- A *monoid* $\mathcal{M}$ is a set M with a unit element $e \in M$ and a binary operator $*\colon M \times M \to M$ such that

  1. e is the unit for $*$:

  $$\forall m \in M.\ e * m = m = m * e, \text{ and}$$

  2. $*$ is associative:

  $$\forall m, m', m'' \in M.\ \ m * (m' * m'') = (m * m') * m''.$$

- Each m in M represents a resource.
- The unit element e denotes the "empty" resource.
- The $*$ operator combines resources.

Examples

$$\begin{aligned} \mathsf{Sentences} &\stackrel{\text{def}}{=} \mathrm{ASCII}^* \\ \mathsf{Money} &\stackrel{\text{def}}{=} \{0, 1, 2, \dots\} \\ \mathsf{Capabilities} &\stackrel{\text{def}}{=} \{\mathsf{access}(n) \mid n \in \mathsf{Nats}\} \end{aligned}$$

- Sentence Model: $(\mathsf{Sentences}, \epsilon, \cdot)$
- Money Model: $(\mathsf{Money}, 0, +)$
- Capability Model: $(\mathcal{P}(\mathsf{Capabilities}), \emptyset, \cup)$

Questions

Suppose that we have predicates `correct`, and `startWith`(s) and `endWith`(s) for all strings s in the sentence model.

$s' \models$ `correct` : s' is grammatically correct.

$s' \models$ `startWith`(s) : s' starts with s.

$s' \models$ `endWith`(s) : s' ends with s.

Can you express the following?

- This sentence can be split into two so that the first part ends with "sentence" and the second part starts with "can".
- Even if I substitute "semantically." for "grammatically." at the end of this sentence, the sentence still remains correct grammatically.

2. Syntax and Semantics

How to express properties about resources?

Syntax

A formula P is given by the following context free grammar:

$$P ::= A \mid 0 \mid P \circ P \mid P \to P \mid P \leftarrow P$$

Semantics of Formulas

For a given monoid $(M, e, *)$, each formula P expresses a set of resources $m \in M$ that satisfy P, denoted $m \models P$.

$$
\begin{array}{lll}
m \models A & \text{iff} & m \in [\![A]\!] \\
m \models 0 & \text{iff} & m = e \\
m \models P \circ Q & \text{iff} & \exists m_1, m_2 \in M. \\
 & & \quad \text{s.t. } m_1 * m_2 = m \text{ and } m_1 \models P \text{ and } m_2 \models Q \\
m \models P \to Q & \text{iff} & \forall m_1 \in M. \\
 & & \quad \text{if } m_1 \models P, \text{ then } m * m_1 \models Q \\
m \models Q \leftarrow P & \text{iff} & \forall m_1 \in M. \\
 & & \quad \text{if } m_1 \models P, \text{ then } m_1 * m \models Q
\end{array}
$$

Examples from Sentence Model

$s' \models \texttt{correct}$	iff	s' is grammatically correct
$s' \models \texttt{startWith}(s)$	iff	s' starts with s
$s' \models \texttt{endWith}(s)$	iff	s' ends with s
$s' \models \texttt{is}(s)$	iff	s' is s

A sentence s can be split into two so that the first part ends with "sentence" and the second part starts with "can".

$$\texttt{endWith(sentence)} \circ \texttt{startWith(can).}$$

Even if I substitute "semantically." for "grammatically." at the end of this sentence, the sentence still remains correct grammatically.

$$(\texttt{is(semantically.)} \rightarrow \texttt{correct}) \circ \texttt{is(grammatically.)}$$

Examples from Money Model

$$
\begin{array}{lll}
m \models \texttt{canBuy(Computer)} & \text{iff} & m \geq 1000000 \\
m \models \texttt{canBuy(PS)} & \text{iff} & m \geq 320000 \\
m \models \texttt{lotto} & \text{iff} & m = 100,000,000 \\
m \models \texttt{poor} & \text{iff} & m \leq 200,000,010
\end{array}
$$

I have enough money to buy both a computer and PS2.

$$\texttt{canBuy(PS)} \circ \texttt{canBuy(Computer)}.$$

Even if I hit the jackpot twice in lotto, I am still poor.

$$\texttt{lotto} \rightarrow (\texttt{lotto} \rightarrow \texttt{poor}).$$

Examples from Capability Model

$$
\begin{array}{lll}
m \models \texttt{capability}(A) & \text{iff} & m = \{\mathsf{access}(0)\} \\
m \models \texttt{crash} & \text{iff} & \{\mathsf{access}(0), \mathsf{access}(1)\} \subseteq m \\
m \models \texttt{cheat} & \text{iff} & \mathsf{access}(0) \notin m \text{ and } \mathsf{access}(2) \notin m
\end{array}
$$

If I obtain the capacities of the user A in addition to what I already have, I can crash the machine.

$$\texttt{capability}(A) \rightarrow \texttt{crash}$$

I have capabilities enough to crash the machine; and when I throw away some of the capabilities for crashing the machine, I can cheat the system administrator.

$$\texttt{crash} \circ \texttt{cheat}$$

3. Proof Rules and Soundness

How to reason about formulas, that is, properties about resources?

Contexts and Sequents

- A *context*, denoted Γ, is a finite *sequence* of formulas.
- A context "$P_0, P_1, \ldots, P_n$" means $P_1 \circ \ldots \circ P_n$.
- A *sequent* is a pair of context Γ and formula P, denoted $\Gamma \vdash P$.
- For a given monoid $(M, e, *)$, a sequent $\Gamma \vdash P$ means that

  for all $m \in M$, if $m \models \Gamma$, then $m \models P$.
- We will consider proof rules about sequents.

Natural Deduction

Our proof system is in the style of natural deduction. That is, the rules have one of the following three forms:

1. a proof rule for using assumption,

$$\frac{}{P \vdash P}\ \mathrm{Id}$$

2. introduction and elimination rules for each connective, and

$$\frac{\Gamma_1 \vdash P_1 \quad \ldots \quad \Gamma_n \vdash P_n}{\Gamma \vdash P \mathbin{?} P'}\ ?I \qquad \frac{\Gamma \vdash P \mathbin{?} P' \quad \Gamma_1 \vdash P_1 \quad \ldots \quad \Gamma_n \vdash P_n}{\Gamma'' \vdash P''}\ ?E$$

3. structural rules.

$$\frac{\Gamma \vdash P}{\Gamma' \vdash P}$$

A *proof* is a tree whose leaf nodes use the Id rule, and whose internal nodes use one of the other rules.

Introduction and Elimination Rules for →, ←, and ∘

$$\frac{\Gamma, P \vdash Q}{\Gamma \vdash P \to Q} \qquad \frac{P, \Gamma \vdash Q}{\Gamma \vdash Q \leftarrow P} \qquad \frac{\Gamma \vdash P \quad \Gamma' \vdash Q}{\Gamma, \Gamma' \vdash P \circ Q}$$

$$\frac{\Gamma \vdash P \to Q \quad \Gamma' \vdash P}{\Gamma, \Gamma' \vdash Q} \qquad \frac{\Gamma \vdash Q \leftarrow P \quad \Gamma' \vdash P}{\Gamma', \Gamma \vdash Q} \qquad \frac{\Gamma \vdash P \circ Q \quad \Gamma', P, Q, \Gamma'' \vdash R}{\Gamma', \Gamma, \Gamma'' \vdash R}$$

Example Derivation:

$$\dfrac{\dfrac{}{R \circ P \vdash R \circ P} \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{R \to (P \to Q) \vdash R \to (P \to Q)} \quad \dfrac{}{R \vdash R}}{R \to (P \to Q),\ R \vdash P \to Q} \to E \quad \dfrac{}{P \vdash P}}{R \to (P \to Q),\ R,\ P \vdash Q} \to E}{R \to (P \to Q),\ R \circ P \vdash Q} \circ E}{R \to (P \to Q) \vdash R \circ P \to Q} \to I$$

Structural Rules and Substructural Logic

Permutation

$$\frac{\Gamma, Q, P, \Gamma' \vdash R}{\Gamma, P, Q, \Gamma' \vdash R}$$

Weakening

$$\frac{\Gamma, \Gamma' \vdash R}{\Gamma, P, \Gamma' \vdash R}$$

Contraction

$$\frac{\Gamma, P, P, \Gamma' \vdash R}{\Gamma, P, \Gamma' \vdash R}$$

We often do not include some structural rules, depending on the resource monoid of interest.

When the structural rules are restricted in a proof system, the system is called *substructural logic*.

	Permutation	Weakening	Contraction
Ordered Linear Logic	No	No	No
Linear Logic (BCI)	Yes	No	No
Relevant Logic	Yes	No	Yes
Affine Logic	Yes	Yes	No
Classical Logic	Yes	Yes	Yes

Soundness

A proof rule

$$\frac{\Gamma_1 \vdash P_1 \ldots \Gamma_n \vdash P_n}{\Gamma \vdash P}$$

is called *sound* for a monoid $\mathcal{M}$ if and only if

> if all of $\Gamma_i \vdash P_i$ are true in $\mathcal{M}$, then $\Gamma \vdash P$ is true in $\mathcal{M}$.

Theorem (Soundness): For all monoids $\mathcal{M}$, the identity rule and the rules for connectives are sound for $\mathcal{M}$.

However, structural rules are sound only for particular monoids.

Structural Rules and the Operator $*$

The soundness of each structural rule depends on the property of the resource-combining operator $*$ in $\mathcal{M} = (M, e, *)$.

1. If $*$ is commutative, then Permutation is sound for $\mathcal{M}$.

2. If $*$ is absorbing, then Weakening is sound for $\mathcal{M}$.

   $*$ is *absorbing* iff for all $m, m' \in M$,
   $$m * m' = m' * m = m.$$

3. If $*$ is idempotent, then Contraction is sound for $\mathcal{M}$.

   $*$ is *idempotent* iff for all $m \in M$, $m * m$ is equal to m

Properties of $*$ for Each Model

Model	Comm.	Absorb.	Idem.	Proof System
Sentences Model	No	No	No	Ordered Linear Logic
Money Model	Yes	No	No	Linear Logic
Capability Model	Yes	No	Yes	Relevant Logic
Trivial Model	Yes	Yes	Yes	Classical Logic

Example Proofs

$$\dfrac{\dfrac{}{\texttt{startWith(I)} \vdash \texttt{startWith(I)}} \quad \dfrac{}{\texttt{endWith(happy)} \vdash \texttt{endWith(happy)}}}{\texttt{startWith(I)}, \texttt{endWith(happy)} \vdash \texttt{startWith(I)} \circ \texttt{endWith(happy)}}$$

$$\dfrac{\dfrac{\dfrac{}{\texttt{canBuy(Computer)} \vdash \texttt{canBuy(Computer)}} \quad \dfrac{}{\texttt{canBuy(PS)} \vdash \texttt{canBuy(PS)}}}{\texttt{canBuy(Computer)}, \texttt{canBuy(PS)} \vdash \texttt{canBuy(Computer)} \circ \texttt{canBuy(PS)}}}{\texttt{canBuy(PS)}, \texttt{canBuy(Computer)} \vdash \texttt{canBuy(Computer)} \circ \texttt{canBuy(PS)}}\ \mathrm{P}$$

$$\dfrac{\dfrac{\dfrac{}{\texttt{cheat} \circ \texttt{cheat}} \quad \dfrac{}{\texttt{cheat} \vdash \texttt{cheat}}}{\texttt{cheat}, \texttt{cheat} \vdash \texttt{cheat} \circ \texttt{cheat}}}{\texttt{cheat} \vdash \texttt{cheat} \circ \texttt{cheat}}\ \mathrm{C}$$

But note that $\texttt{cheat}, \texttt{crash} \vdash \texttt{cheat} \circ \texttt{cheat}$ does not hold.

Conclusion

- Substructural logic is a proof system where structural rules are used in a restricted way.
- A connective in substructural logic is resource-sensitive: $\circ$ denotes a resource splitting, and $\rightarrow$ the addition of resources.
- The interpretation of substructural logic by monoid in this slide is a special case of a more general model. For the more general model theory, look at
  - "Introduction to Substructural Logics" by Greg Restall
  - "Possible Worlds and Resources: Semantics of BI" by Pym, O'Hearn, and Yang