Lecture 2: Introducing Definitions

Dale Miller

September 2000

Capturing may and must quantification

MAY "There exists a transition such that . . ." corresponds to "There exists a proof of . . .". (adequacy theorem)

MUST "For every transition . . ." corresponds to ???.

Notions such as *simulation*, *bisimulation*, and *deadlock freeness* all incorporate MUST quantification.

In this rarefied setting of proof theory, many other things are achieved using MUST quantification (as we shall see).

Definitions defined

A *definition* is a finite set of definition *clause*

$$\forall\bar{x}[p_1(\bar{t}_1) \triangleq B_1] \quad \ldots \quad \forall\bar{x}[p_n(\bar{t}_n) \triangleq B_n] \quad (N \geq 0)$$

For $i = 1, \ldots, n$,

- p_i is a predicate constant,
- every free variable of the formula B_i is also free in at least on term in the list $\bar{I}_i$ of terms, and
- all variables free in $\bar{t}_i$ are contained in the list $\bar{x}_i$.

The formula B_i is the *body* and $p_i(\bar{t}_i)$ is the *head* of the i^{th} clause.

The predicate symbols $p_1, \ldots, p_n$ are not distinct predicates: definitions act to define predicates by mutual recursion.

The symbol $\triangleq$ is not a logical connective: it is used just to denote definitional clauses.

Right introduction for defined atoms

Left and right introduction rules for atomic formulas will be given for a fixed definition and equality theory.

$$\frac{\Delta \longrightarrow B\theta}{\Delta \longrightarrow A} \; \mathit{def}\mathcal{R}, \text{ where } A = H\theta \text{ for some clause } \forall\bar{x}.[H \triangleq B].$$

If we think of a definition as a logic program, then this rule is essentially the same as *backchaining*.

Notice that (reading from bottom up) the atom is replace by *some* ($\exists$) body of a matching clause.

Notice that *matching* is used to select a clause from a definition.

Left introduction for defined atoms

$$\frac{\{B\theta, \Delta\theta \longrightarrow C\theta \mid \theta \in csu(A, H) \text{ for some clause } \forall \bar{x}.[H \triangleq B]\}}{A, \Delta \longrightarrow C}\ def\mathcal{L}.$$

The variables $\bar{x}$ need to be chosen so that they are not free in any formula of the lower sequent. This rule is due to [Eriksson 91].

Notice that (reading from bottom up) the atom is replace by *all* ($\forall$) bodies of a matching clauses.

The set of premises can be empty, finite, or infinite since definitions since the set $csu(A, H)$ can be infinite. In some theories, CSUs are not effectively computable.

While the theory of definitions is this general case, we shall only use this left rule when CSUs can be replaced with MGUs (most general unifiers).

Notice that *unification* is used to select clauses from a definition.

Examples: $1 + 2 = 3$ and $1 + 2 \neq 1$

Define addition on a simple encoding of natural numbers.

$$\begin{array}{rcl} sum\ z\ N\ N & \triangleq & \top. \\ sum\ (s\ N)\ M\ (s\ P) & \triangleq & sum\ N\ M\ P. \end{array}$$

We can prove that $1 + 2 = 3$

$$\dfrac{\dfrac{\longrightarrow \top}{\longrightarrow sum\ z\ (s\ (s\ z))\ (s\ (s\ z))}\ def\mathcal{R}}{\longrightarrow sum\ (s\ z)\ (s\ (s\ z))\ (s\ (s\ (s\ z)))}\ def\mathcal{R}$$

and that $1 + 2 \neq 1$.

$$\dfrac{\dfrac{}{sum\ z\ (s\ (s\ z))\ z \longrightarrow}\ def\mathcal{L}}{sum\ (s\ z)\ (s\ (s\ z))\ (s\ z) \longrightarrow}\ def\mathcal{L}$$

More generally, consider (where n and m are eigen-variables):

$$\dfrac{\Gamma(n, n), \top \longrightarrow G(n, n)}{\Gamma(n, m), sum\ z\ n\ m \longrightarrow G(n, m)}\ def\mathcal{L}$$

Example: evaluation of a conditional

Consider defining some rules for a conditional (if) in a functional programming language.

$$\vdots$$

$$\begin{array}{rcl} (if\ B\ M\ N) \Downarrow V & \triangleq & B \Downarrow \mathit{true} \wedge M \Downarrow V. \\ (if\ B\ M\ N) \Downarrow V & \triangleq & B \Downarrow \mathit{false} \wedge N \Downarrow V. \end{array}$$

$$\vdots$$

Now consider the following fragment of a proof

$$\dfrac{\dfrac{}{B \Downarrow \mathit{true} \wedge M \Downarrow V \longrightarrow M \Downarrow V}\ \mathit{initial} \qquad \dfrac{}{B \Downarrow \mathit{false} \wedge M \Downarrow V \longrightarrow M \Downarrow V}\ \mathit{initial}}{(if\ B\ M\ M) \Downarrow V \longrightarrow M \Downarrow V}\ \mathit{def}\mathcal{L}$$

Roles for $def\mathcal{R}$ and $def\mathcal{L}$

$$\dfrac{\vdots}{\longrightarrow A}\ def\mathcal{R}$$ corresponds to *backchaining*.

$$\dfrac{\vdots}{A \longrightarrow}\ def\mathcal{L}$$ corresponds to *finite failure*.

$$\dfrac{\vdots}{A \longrightarrow B}\ def\mathcal{L} + def\mathcal{R}$$ corresponds to *simulation*.

If a defintion is thought of as logic program, by replacing $\triangleq$ with $\subset$ (reverse implication), then $def\mathcal{R}$ can be modeled correctly within sequent calculus. $def\mathcal{L}$ cannot be so directly modeled. The case analysis of $def\mathcal{L}$ is something new.

Cut-elimination for defintions

Without restrictions, cut-elimination fails for definitions. Consider, for example, the definition (p if and only if not p):

$$p \triangleq p \supset \bot \ .$$

Let Ξ_1 be the proof

$$\cfrac{\cfrac{\cfrac{p \longrightarrow p \qquad \bot \longrightarrow \bot}{p, p \supset \bot \longrightarrow \bot}\ \supset\mathcal{L}}{p, p \longrightarrow \bot}\ def\mathcal{L}}{p \longrightarrow \bot}\ Contraction$$

and let Ξ_2 be the proof

$$\cfrac{\cfrac{\Xi_1}{\longrightarrow p \supset \bot}\ \supset\mathcal{R}}{\longrightarrow p}\ def\mathcal{R}.$$

Using cut with Ξ_1 and Ξ_2, we have a proof of $\longrightarrow \bot$. There is no cut-free proof of this sequent.

Restrictions on definitions

The literature contains three ways to restriction definitions so that cut-elimination can hold.

1. Do not allow the body of definitions to contain implications. This is a rather strong restriction, but corresponds to Horn clauses. [Schroeder-Heister]
2. Remove contraction, which moves us away from intuitionistic logic to linear or relevant logics. [Girard, Schroeder-Heister]
3. Give predicates and formulas a level and require definitions to be stratified [McDowell&Miller]. This is the choice we make here.

Level restriction on definitions

Each predicate symbol p in the logic has associated with it a natural number $lvl(p)$, the *level* of the predicate. A level is then given to arbitrary formulas as follows:

1. $lvl(p\,\bar{t}) = lvl(p)$
2. $lvl(\perp) = lvl(\top) = 0$
3. $lvl(B \wedge C) = lvl(B \vee C) = \max(lvl(B), lvl(C))$
4. $lvl(B \supset C) = \max(lvl(B) + 1, lvl(C))$
5. $lvl(\forall x.B) = lvl(\exists x.B) = lvl(B)$.

We now require that for every definitional clause $\forall\bar{x}[p\,\bar{t} \triangleq B]$, $lvl(B) \leq lvl(p\,\bar{t})$.

Notice that there is no way to assign a level to p so that

$$p \triangleq p \supset \perp .$$

satisfies this level-restriction. Definitions without implications in their body can immediately satisfy this level restriction.

Abstract transition systems

The triple $\mathcal{T} = (\Lambda, S, \delta)$ is an *abstract transition system* (*ats*) if

1. Λ is a non-empty set of *actions*,
2. S is a non-empty set of *states* (Λ and S are assumed to be disjoint), and
3. $\delta \subseteq S \times \Lambda \times S$.

We write $p \xrightarrow{a} q$ if $(p, a, q) \in \delta$.

If $w \in \Lambda^*$ then we write $p \stackrel{w}{\Longrightarrow} q$ to mean that p makes a transition to q along a path of actions given by w: thus $p \stackrel{\epsilon}{\Longrightarrow} p$ and if $p \xrightarrow{a} r$ and $r \stackrel{w}{\Longrightarrow} q$ then $p \stackrel{aw}{\Longrightarrow} q$.

For a state p, define $\langle\langle p \rangle\rangle = \{(a, q) \mid (p, a, q) \in \delta\}$.

The ats $\mathcal{T}$ is

1. *finitely branching* if, for each p, the set $\langle\langle p \rangle\rangle$ is finite.
2. *determinate* if for every state p and every action a, the set $\{q \mid (p, a, q) \in \delta\}$ is either empty or a singleton.
3. *noetherian* if it contains no infinite paths.

Encoding an ats as a definition

Given an ats $\mathcal{T} = (\Lambda, S, \delta)$, we encode the members of Λ and S as individual constants in a first order logic.

The transition relation δ is encoded as a definition, called $ats(\delta)$, and uses a single ternary predicate *one*, which has one clause

$$one(p, a, q) \triangleq \top$$

for each triple $\langle p, a, q \rangle \in \delta$.

The definition *path* is given by the following two clauses.

$$\begin{array}{rccl} \forall P[& multi(P, nil, P) & \triangleq & \top] \\ \forall A, P, Q, W[& multi(P, A :: W, Q) & \triangleq & \exists R(one(P, A, R) \wedge multi(R, W, Q))] \end{array}$$

CCS as an abstraction transition system

Let $\mathcal{A}$ be a fixed, finite set of actions, let $\bar{\mathcal{A}}$ be the set of complementary actions $\{\bar{a} \mid a \in \mathcal{A}\}$ and let $\Lambda = \mathcal{A} \cup \bar{\mathcal{A}} \cup \{\tau\}$.

Let S be the set of CCS expression

$$p \ ::= \ 0 \ \Big| \ x \ \Big| \ a.p \ \Big| \ p + p \ \Big| \ p \,|\, p \ \Big| \ \mu_x p,$$

where x ranges over process variables and a ranges over Λ.

The relation δ is given by the following proof rules.

$$\frac{}{a.p \xrightarrow{a} p} \qquad \frac{p \xrightarrow{a} q}{p \,|\, r \xrightarrow{a} q \,|\, r} \qquad \frac{p \xrightarrow{a} q}{r \,|\, p \xrightarrow{a} r \,|\, q} \qquad \frac{p[\mu_x p/x] \xrightarrow{a} q}{\mu_x p \xrightarrow{a} q}$$

$$\frac{p \xrightarrow{a} q}{p + r \xrightarrow{a} q} \qquad \frac{r \xrightarrow{a} q}{p + r \xrightarrow{a} q} \qquad \frac{p \xrightarrow{a} r \quad q \xrightarrow{b} s}{p \,|\, q \xrightarrow{\tau} r \,|\, s} \text{ where } b = \bar{a} \text{ or } a = \bar{b}$$

The type of the μ constructor must be sometime like $(\mathit{process} \rightarrow \mathit{process}) \rightarrow \mathit{process}$.

The definition $ccs(\mathcal{A})$

$$
\begin{array}{rccl}
\forall A, P\, [& one(A.P, A, P) & \triangleq & \top] \\
\forall A, P, Q, R\, [& one(P \mid R, A, Q \mid R) & \triangleq & one(P, A, Q)] \\
\forall A, P, Q, R\, [& one(R \mid P, A, R \mid Q) & \triangleq & one(P, A, Q)] \\
\forall A, P, Q, R\, [& one(P + R, A, Q) & \triangleq & one(P, A, Q)] \\
\forall A, P, Q, R\, [& one(P + R, A, Q) & \triangleq & one(R, A, Q)] \\
\forall A, P, Q\, [& one(\mu M, A, Q) & \triangleq & one(M(\mu M), A, Q)]
\end{array}
$$

$$\forall P, Q, R, S\; [one(P|Q, \tau, R|S) \triangleq \exists A, B(comp(A, B) \wedge one(P, A, R) \wedge one(Q, B, S))]$$

For each $a \in \mathcal{A}$, the two clauses $comp(a, \bar{a}) \triangleq \top$ and $comp(\bar{a}, a) \triangleq \top$.

Simulation and bisimulation

A relation $\mathcal{R} \subseteq \mathcal{S} \times \mathcal{S}$ is a *simulation* between p and q if and only if for every transition $p \xrightarrow{a} p'$, there exists a transition $q \xrightarrow{a} q'$, such that $p'\mathcal{R}q'$.

The union of all simulations in an ats is also a simulation in the ats and we denote it by $\sqsubseteq$; that is, $p \sqsubseteq q$ (read "q simulates p") holds if and only if there exists a simulation $\mathcal{R}$ such that $p\mathcal{R}q$.

If $p \sqsubseteq q$ and $q \sqsubseteq p$ both hold, then p and q are *similar*.

A relation $\mathcal{R}$ is a *bisimulation* between p and q if and only if for every transition $p \xrightarrow{a} p'$, there exists a transition $q \xrightarrow{a} q'$ such that $p'\mathcal{R}q'$, and for every transition $q \xrightarrow{a} q'$, there exists a transition $p \xrightarrow{a} p'$ such that $q'\mathcal{R}p'$.

The union of all bisimulations in an ats is also a bisimulation in the ats and we denote it by $\equiv$; that is, $p \equiv q$ (read "p is *bisimilar* to q") holds if and only if there exists a bisimulation $\mathcal{R}$ such that $p\mathcal{R}q$.

Similar is not bisimilar

It is well-known that bisimilarity implies similarity but not vice-versa.

Consider the transition system with one label a, states $\{p_1, p_2, p_3, p_4, q_1, q_2, q_3\}$, and transitions $p_1 \xrightarrow{a} p_2, p_2 \xrightarrow{a} p_3, p_1 \xrightarrow{a} p_4, q_1 \xrightarrow{a} q_2, q_2 \xrightarrow{a} q_3$. The relations

$$\{(p_1, q_1), (p_2, q_2), (p_3, q_3), (p_4, q_2)\} \text{ and } \{(q_1, p_1), (q_2, p_2), (q_3, p_3)\}$$

witness the fact that q_1 simulates p_1 and p_1 simulates q_1, respectively. It is easy to check, however, that there is no bisimulation that contains the pair (p_1, q_1).

The definition *sims*

$$
\begin{array}{lrcl}
\forall P, Q\ [& sim(P,Q) & \triangleq & \forall A \forall P'.\ one(P,A,P') \supset \\
& & & \qquad \exists Q'.\ one(Q,A,Q') \wedge sim(P',Q')] \\
\forall P, Q\ [& bisim(P,Q) & \triangleq & [\forall A \forall P'.\ one(P,A,P') \supset \\
& & & \qquad \exists Q'.\ one(Q,A,Q') \wedge bisim(P',Q')] \wedge \\
& & & [\forall A \forall Q'.\ one(Q,A,Q') \supset \\
& & & \qquad \exists P'.\ one(P,A,P') \wedge bisim(Q',P')]]
\end{array}
$$

The clauses in this definition are not a Horn clauses.

This definition is stratified with predicate $one(\cdot,\cdot,\cdot)$ at a level lower than $sim(\cdot,\cdot)$ and $bisim(\cdot,\cdot)$.

Simulation as an inference rule

Let p and q be two states such that

$$\{(a_1, p_1), \ldots, (a_n, p_n)\} = \langle\langle p \rangle\rangle \quad \text{and} \quad \{(a_1, q_1), \ldots, (a_n, q_n)\} \subseteq \langle\langle q \rangle\rangle$$

(Notice that we are assuming the ats to be finitely branching.)

A proof of $\longrightarrow \mathit{sim}(p, q)$ can always be built as follows:

$$
\dfrac{\dfrac{\dfrac{\cdots \qquad \dfrac{\dfrac{}{\longrightarrow \mathit{one}(q, a_i, q_i)}\,\mathit{def}\mathcal{R} \qquad \dfrac{\vdots}{\longrightarrow \mathit{sim}(p_i, q_i)}}{\longrightarrow \exists Q'.\ \mathit{one}(q, a_i, Q') \wedge \mathit{sim}(p_i, Q')}\,\exists\mathcal{R}, \wedge\mathcal{R} \qquad \cdots}{\mathit{one}(p, A, P') \longrightarrow \exists Q'.\ \mathit{one}(q, A, Q') \wedge \mathit{sim}(P', Q')}\,\mathit{def}\mathcal{L}}{\longrightarrow \forall A \forall P'.\ \mathit{one}(p, A, P') \supset \exists Q'.\ \mathit{one}(q, A, Q') \wedge \mathit{sim}(P', Q')}\,\forall\mathcal{R}, \supset\mathcal{R}}{\longrightarrow \mathit{sim}(p, q)}\,\mathit{def}\mathcal{R}
$$

Simulation as a derived inference rule

The proceeding slide can be used to justify the following derived inference rule.

$$\frac{\longrightarrow sim(p_1, q_1) \quad \ldots \quad \longrightarrow sim(p_n, q_n)}{\longrightarrow sim(p, q)}$$

where p and q are two states such that

$$\{(a_1, p_1), \ldots, (a_n, p_n)\} = \langle\langle p \rangle\rangle \quad \text{and} \quad \{(a_1, q_1), \ldots, (a_n, q_n)\} \subseteq \langle\langle q \rangle\rangle$$

Notice that to determine simulation, *no* other inference rule is need, not even the *initial* inference rule. This rule has no premises if $\langle\langle p \rangle\rangle$ is empty.

Of course, if the transition is not noetherian then (finite) proofs might not exist. One can consider infinite proofs, but that gives up too much.

Example: computing max

Let the predicate a be given by a table, for example, as:

$$
\begin{aligned}
a\ (s\ z) &\triangleq \top. \\
a\ (s\ (s\ (s\ z))) &\triangleq \top. \\
a\ z &\triangleq \top.
\end{aligned}
$$

How can you specify the predicate $maxa\ N$ so that this is provable if and only if N is the large natural number argument for a.

This is impossible if logic programs are given (not definitions). It is only possible if we use $defL$. Why?

This specification will work. And it's the natural one!

$$
\begin{aligned}
maxa\ N &\triangleq (a\ N) \wedge \forall x(a\ x \supset x \leq N). \\
z \leq N &\triangleq \top. \\
(s\ N) \leq (s\ M) &\triangleq N \leq M.
\end{aligned}
$$

Example: membership in all paths

Assume that an ats is specified as before, along with the *multi* predicate.

Specify: a defintion for *somepath* $p\ a\ q$ that is provable if and only if there is a path from p to q that contains the label a.

$$\mathit{somepath}\ p\ a\ q \triangleq \exists w\ [\mathit{multi}\ p\ w\ q \wedge \mathit{member}\ a\ w]$$

where *member* is defined by the usual Horn clause program.

Specify: a definition for *allpath* $p\ a\ q$ that is provable if and only if every path from p to q contains the label a.

$$\mathit{allpath}\ p\ a\ q \triangleq \forall w\ [\mathit{multi}\ p\ w\ q \supset \mathit{member}\ a\ w]$$

The predicate *somepath* can be written using logic programs but the predicate *allpath* cannot be written using logic programs ($\mathit{def}\mathcal{L}$ is necessary). This example again illustrates the difference between may and must behaviors.

Definitions as fixed points

A definition $\mathcal{D}$ can be seen as an operator $T_{\mathcal{D}}$ from interpretation I to interpretation $T_{\mathcal{D}}(I)$, so that

$$T_{\mathcal{D}}(I) = \{A \mid \text{there is an instance } A \triangleq B \text{ of a clause in } \mathcal{D} \text{ such that } B \text{ is true in } I\}$$

Statements provable using $\mathcal{D}$ are true in all fixed points of $T_{\mathcal{D}}$.

Finite and noetherian transition systems have unique fixed points.

Consider the non-noetherian ats with δ being $\{\langle p, a, p\rangle, \langle q, a, q\rangle\}$.

There are two fixed points for *bisim*, namely, $\{\}$ and $\{\langle p, q\rangle\}$. Thus there is no proof of $\mathit{bisim}(p, q)$. But, $p \equiv q$.

Simulation and bisimulation for noetherian ats

Theorem. If a noetherian ats is encoded as a definition, then

1. $p \sqsubseteq q$ if and only if $\vdash \mathit{sim}(p, q)$.
2. $p \equiv q$ if and only if $\vdash \mathit{bisim}(p, q)$.

Let *finite CCS* expressions be CCS expressions without recursion (without the μ operator). Then finite CCS expressions form a noetherian ats and the above theorem holds for it (even the encoding of one-step transitions is different).

Initial can be derived

In most sequent systems, the initial sequent $B \longrightarrow B$ can be restricted so that B is an atomic formula.

$$\dfrac{\dfrac{B \longrightarrow B}{B \wedge C \longrightarrow B}\ \wedge\mathcal{L} \qquad \dfrac{C \longrightarrow C}{B \wedge C \longrightarrow C}\ \wedge\mathcal{L}}{B \wedge C \longrightarrow B \wedge C}\ \wedge\mathcal{R}$$

$$\dfrac{\dfrac{B \longrightarrow B \quad C \longrightarrow C}{B \supset C, B \longrightarrow C}\ \supset\mathcal{L}}{B \supset C \longrightarrow B \supset C}\ \supset\mathcal{R}$$

If a definition is not recursive, even atomic initial sequents can be derived.

$$\dfrac{\dfrac{B_1 \longrightarrow B_1}{B_1 \longrightarrow A}\ def\mathcal{R} \quad \vdots \quad \dfrac{B_n \longrightarrow B_n}{B_n \longrightarrow A}\ def\mathcal{R}}{A \longrightarrow A}\ def\mathcal{L}$$

Definitions are not enough

While definitions can be used to enumerate and explore a search space, they can neither observe nor use invariances over a computation. There is little they can do with infinite behavior.

Some properties that can be proved using only definitions:

$$\vdash \forall P \forall Q.\ bisim(P, Q) \supset bisim(Q, P)$$

$$\vdash \forall P \forall Q \forall R.\ bisim(P, Q) \wedge bisim(Q, R) \supset bisim(P, R)$$

And for CCS:

$$\vdash \forall A \forall P \forall Q.\ bisim(P, Q) \supset bisim(A.P, A.Q)$$

But many properties cannot be proved:

$$\vdash \forall P.\ bisim(P, P)$$