Choice quantification in process algebra
Luttik, S.P.

Citation for published version (APA):

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: http://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.
A correspondence between \textit{pCRL} and first-order logic

The language \textit{pCRL} is parametrised with a data algebra \( \mathbf{D} \). As explained in Section 3.4, its expressions correspond with certain infinitely branching trees. Which trees correspond with \textit{pCRL} expressions depends in part on \( \mathbf{D} \). For instance, for the infinitely branching tree pictured in Figure 3.2 on p. 39 we need that the domain of \( \mathbf{D} \) consists of integers, and that \( \mathbf{D} \) has a relation \( \leq \) or a function \( \lfloor \cdot \rfloor \) that computes the absolute value.

It is to be expected that the validity in \( T_{\mathbf{D}}(\mathcal{A}) \) of \textit{pCRL} equations also depends in some way on \( \mathbf{D} \). For instance, if \( d \) and \( e \) are closed data expressions and \( a \) is a unary parametrised action, then

\[
T_{\mathbf{D}}(\mathcal{A}) \models a(d) \approx a(e) \text{ if, and only if, } \mathbf{D} \models d \approx e.
\]

Also, if \( b \) is a closed Boolean expression and \( p \) and \( q \) are closed \textit{pCRL} expressions, then

\[
T_{\mathbf{D}}(\mathcal{A}) \models \begin{cases} 
p \land b \land q \approx p & \text{if } \mathbf{D} \models b \approx \top; \text{ and} \\
p \land b \land q \approx q & \text{if } \mathbf{D} \models b \approx \bot. 
\end{cases}
\]

And even if the validity of data equations and Boolean equation in \( \mathbf{D} \) is decidable, the validity of a \textit{pCRL} equation in \( T_{\mathbf{D}}(\mathcal{A}) \) may still be undecidable.

**Example 4.1** Suppose that we take as data the natural numbers with Kleene's \( T \)-predicate: if \( z \) is the encoding (i.e., Gödel number) of Turing machine \( Z \), then

\[
T(z,x,y) = \top \text{ if, and only if, } y \text{ encodes a computation}^{1} \text{ of } Z \text{ on } x.^{2}
\]

Kleene's \( T \)-predicate is known to be primitive recursive. Now, consider the \textit{pCRL} expression

\[
p(z,x) = \sum_y c \land T(z,x,y) \land \delta, \text{ where } c \text{ is any closed action expression.}
\]

---

1A computation is a sequence of pairs consisting of a state and a string that represents the contents of the tape, such that the last state in the sequence is a final state.

2In the recursion theory literature (e.g., Davis, 1982; Rogers, Jr., 1992) one finds the predicates \( T_n(z,x_1,\ldots,x_n,y) \), where \( Z \) takes the sequence \( x_1,\ldots,x_n \) as input; we shall only use \( T_1 \) and drop the subscript.
If $Z$ has a successful computation on input $x$, then $T_D(A) \models p(z, x) \approx c$; otherwise $T_D(A) \models p(z, x) \approx \delta$. So $p(z, x) \approx c$ holds in $T_D(A)$ if, and only if, the first-order formula

$$(\exists y) T(z, x, y)$$

holds in $D$. This formula defines an undecidable relation on the natural numbers -- it corresponds to the halting problem (Turing, 1936) — so validity in $T_D(A)$ is undecidable.

Although existential quantifiers are not part of our definition of Boolean expressions, they pop up when we consider validity in $T_D(A)$. Example 4.1 shows that the validity in $T_D(A)$ of a pCRL equation may be undecidable if there exist undecidable first-order assertions about the data. We shall see below that it is necessary and sufficient for the decidability of validity in $T_D(A)$ that all first-order assertions about the data are decidable.

The set $\Phi$ of first-order formulas is generated by

$$\varphi ::= r(d_1, \ldots, d_n) \mid \neg \varphi \mid \varphi \lor \varphi \mid (\exists x)\varphi,$$

where $d_1, \ldots, d_n$ are data expressions, $r$ is a relation symbol of arity $n$, and $x$ is a variable. The construct $(\exists x)$ binds the variable $x$ in its argument; we adopt Convention 3.7 also for first-order formulas. For a given valuation $\nu : X \rightarrow D$ we define the satisfaction relation $D, \nu \models \varphi$ inductively as follows:

1. $D, \nu \models r(d_1, \ldots, d_n)$ if, and only if, $R(\nu(d_1), \ldots, \nu(d_n)) = T$, where $R$ is the $n$-ary relation of $D$ corresponding to the relation symbol $r$;

2. $D, \nu \models \neg \varphi$ if, and only if, $D, \nu \not\models \varphi$;

3. $D, \nu \models \varphi \lor \psi$ if, and only if, $D, \nu \models \varphi$ or $D, \nu \models \psi$; and

4. $D, \nu \models (\exists x)\varphi$ if, and only if, there exists $d \in D$ such that $D, \nu[x := d] \models \varphi$, where $\nu[x := d]$ is the valuation such that

$$\nu[x := d](y) = \begin{cases} d & \text{if } y = x; \text{ and} \\ \nu(y) & \text{otherwise.} \end{cases}$$

If $D, \nu \models \varphi$ for all valuations $\nu$, then we write $D \models \varphi$. The first-order theory of $D$ is the set of all formulas $\varphi$ such that $D \models \varphi$.

We also define the pCRL theory of $D$, as the set of all pCRL equations $p \approx q$ such that $T_D(A) \models p \approx q$. We shall reveal the following intimate relationship between the pCRL theory of $D$ and the first-order theory of $D$:

The pCRL theory of $D$ and the first-order theory of $D$ are recursively isomorphic.

That is, there exists a recursive bijection between both theories (see Rogers, Jr., 1992). To prove this, it is by a theorem of Myhill (1955) enough to show that the pCRL theory of $D$ and the first-order theory of $D$ have the same degree of unsolvability with respect to one-one reducibility (Rogers, Jr., 1992). That is, it suffices to define two one-one recursive functions:
1. a one-one recursive function \( \phi : \mathcal{P} \times \mathcal{P} \to \Phi \) such that for every valuation \( \nu \)
\[
\mathcal{T}_D(A), \nu \models p \approx q \text{ if, and only if, } \mathcal{D}, \nu \models \phi(p, q); \text{ and }
\]
2. a one-one recursive function \( \eta : \Phi \to \mathcal{P} \times \mathcal{P} \) such that for every valuation \( \nu \)
\[
\mathcal{D}, \nu \models \varphi \text{ if, and only if, } \mathcal{T}_D(A), \nu \models p \approx q, \text{ where } \eta(\varphi) = \langle p, q \rangle.
\]
The function \( \phi \) will be defined in Section 4.2 (see Theorem 4.10). The function \( \eta \) will be defined in Section 4.3 (see Theorem 4.17). First, however, it is convenient to devote a preliminary section on discussing the precise connection between the Boolean expressions used as conditions in pCRL expressions, and certain first-order formulas.

### 4.1 Boolean expressions and open first-order formulas

Following, e.g., Shoenfield (1967) and Chang and Keisler (1990), we call a first-order formula is open if it contains no quantifiers. Syntactically, every open first-order formula is also a Boolean expression, and the following proposition provides the semantical justification for this ambiguity.

**Proposition 4.2** If \( \varphi \) is an open first-order formulas, then
\[
\mathcal{D}, \nu \models \varphi \text{ if, and only if, } \mathcal{D}, \nu \models \varphi \approx \top
\]
for every valuation \( \nu \).

**Proof.** We proceed by induction on the structure of \( \varphi \).

If \( \varphi = r(d_1, \ldots, d_n) \) and \( r \) denotes the \( n \)-ary relation \( R \) of \( \mathcal{D} \), then
\[
\mathcal{D}, \nu \models \varphi \iff R(\nu(d_1), \ldots, \nu(d_n)) = \top \iff \mathcal{D}, \nu \models \varphi \approx \top
\]
If \( \varphi = \neg \psi \), then, according to the definition of \( \nu \) on p. 32,
\[
\nu(\varphi) = \top \text{ if, and only if, } \nu(\psi) \neq \top;
\]
hence, with an application of the induction hypothesis,
\[
\mathcal{D}, \nu \models \varphi \iff \mathcal{D}, \nu \not\models \psi \iff \mathcal{D}, \nu \not\models \psi \approx \top \iff \mathcal{D}, \nu \models \varphi \approx \top.
\]
If \( \varphi = \psi \lor \chi \), then, according to the definition of \( \nu \) on p. 32,
\[
\nu(\varphi) = \top \text{ if, and only if, } \nu(\psi) = \top \text{ or } \nu(\chi) = \top;
\]
hence, with an application of the induction hypothesis,
\[
\mathcal{D}, \nu \models \varphi \iff \mathcal{D}, \nu \models \psi \text{ or } \mathcal{D}, \nu \models \chi
\]
\[
\iff \mathcal{D}, \nu \models \psi \approx \top \text{ or } \mathcal{D}, \nu \models \chi \approx \top \iff \mathcal{D}, \nu \models \varphi \approx \top.
\]
The proof is complete. \( \square \)
Our definition of first-order formula deviates slightly from that of Shoenfield (1967); Shoenfield presupposes a binary relation symbol with a fixed interpretation as equality. The reason for our deviation is that, for the rest of this chapter, it is convenient to have that every open first-order formula is automatically a Boolean expression. whence may be used as a condition in a pCRL expression. If we now add equality as a special requirement on data algebras, then, of course, this property is maintained.

**Definition 4.3** We say that a data algebra \( D \) has *equality* if, among the relations of \( D \), there is a binary relation denoted by the relation symbol \( \text{eq} \) such that for every valuation \( \nu \):

\[
D, \nu \models \begin{cases} 
\text{eq}(x, y) \approx \top & \text{if } \nu(x) = \nu(y); \\
\text{eq}(x, y) \approx \bot & \text{if } \nu(x) \neq \nu(y).
\end{cases}
\]

Note that, syntactically, Boolean expressions are open first-order formula, unless they contain occurrences of the symbols \( \top, \bot \), or \( \land \). But it is well-known that \( \land \) is definable with \( \neg \) and \( \lor \), and with equality as a binary relation in \( D \). \( \top \) and \( \bot \) turn out to be definable as well. We get that every Boolean expression is semantically equivalent to a first-order formula.

**Proposition 4.4** If \( D \) has equality, then for every Boolean expression \( b \) there exists an open first-order formula \( \varphi \) such that \( D \models b \approx \varphi \).

**Proof.** We make three observations.

Firstly, according to Definition 4.3, for every variable \( x \in X \)

\[
D \models \text{eq}(x, x) \approx \top,
\]

so if \( b = \top \), then we can select \( x \in X \) and put \( \varphi = \text{eq}(x, x) \).

Secondly, since \( \neg \top = \bot \) by definition,

\[
D \models \neg \top \approx \bot,
\]

so if \( b = \bot \), then we can put \( \varphi = \neg \text{eq}(x, x) \).

Thirdly, suppose that \( b = \psi \land \chi \) and \( \psi \) and \( \chi \) are open first-order formulas. Then, since \( b \land c = \neg(\neg b \lor \neg c) \) for all \( b, c \in B \),

\[
D \models \psi \land \chi \approx \neg(\neg \psi \lor \neg \chi).
\]

so we can put \( \varphi = \neg(\neg \psi \lor \neg \chi) \).

With these observations the proposition follows by structural induction on \( b \). \( \square \)

For the most part, we shall be working with Boolean expressions modulo semantic equivalence, and with a data algebra that has equality. Then, according to the above proposition, every Boolean expression may be conceived as an open first-order formula: by (4.1)–(4.3) we may interpret occurrences of \( \top, \bot \) and \( \varphi \land \psi \) as abbreviations of \( \text{eq}(x, x) \), \( \neg \top \), and \( \neg(\neg \varphi \lor \neg \psi) \), respectively.

We introduce a few more standard abbreviations: \( \varphi \rightarrow \psi \) abbreviates \( \neg \varphi \lor \psi \); \( \varphi \leftrightarrow \psi \) abbreviates \( (\varphi \rightarrow \psi) \land (\psi \rightarrow \varphi) \); and \( \forall x \varphi \) abbreviates \( \neg(\exists x) \neg \varphi \). Furthermore, if \( m \geq 1 \) and \( n \geq 0 \), then we define the formula \( \bigvee_{m \leq i \leq n} \varphi_i \) inductively as follows:
4.2 The definition of $\phi$

We start with an analysis of when a valuation $\nu$ satisfies $t \preceq u$ in $T_D(A)$, where $t$ and $u$ are ordered tree forms. Our analysis will lead to the definition of a recursive function $\phi_\leq : T_o \times T_o \to \Phi$ such that for all ordered tree forms $t$ and $u$

$$D, \nu \models \phi_\leq(t, u) \text{ if, and only if, } T_D(A), \nu \models t \preceq u.$$ We shall then obtain $\phi$ from $\phi_\leq$ and the function $\theta$ that assigns to every pCRL expression an equivalent ordered tree form.

First, we distinguish cases according to the form of $t$:

Suppose that $t = \delta$. Since $\delta$ is the least element with respect to $\preceq$ in every generalised basic process algebra with deadlock,

$$T_D(A), \nu \models \delta \preceq u. \quad (4.4)$$

Suppose that $t = t' + t''$, then, since an alternative composition is the least upper bound of its components in every generalised basic process algebra with deadlock,

$$T_D(A), \nu \models t' + t'' \preceq u \text{ if, and only if, } T_D(A), \nu \models t' \preceq u, \ t'' \preceq u. \quad (4.5)$$

Suppose $t$ is a simple tree form, say $t = \sum_{\vec{x}} t^* \preceq b \triangleright \delta$. We need some notation: if $\vec{x} = x_1, \ldots, x_n$ is a sequence of variables, and $\vec{d} = d_1, \ldots, d_n$ is a sequence of elements of $D$, then with $[\vec{x} := \vec{d}]$ we shall mean the sequence

$$[x_n := d_n] \cdots [x_1 := d_1].$$

(The inversion is for convenience of notation; e.g., we have, for a sequence of variables $\vec{x} = x_1, \ldots, x_n$, that $\nu_\ell(\sum_{\vec{x}} p) = \sum\{\nu_\ell(p[\vec{x} := \vec{d}]) | \vec{d} = d_1, \ldots, d_n \in D\}$, also if some variable occurs more than once in $\vec{x}$.)

**Lemma 4.5** Suppose that $\vec{x} = x_1, \ldots, x_n$ is a sequence of variables such that $\{\vec{x}\} \cap \text{FV}(u) = \emptyset$; then

$$T_D(A), \nu \models \sum_{\vec{x}} t^* \preceq b \triangleright \delta \preceq u \text{ if, and only if, }$$

$$\text{for all sequences } \vec{d} = d_1, \ldots, d_n \in D$$

$$D, \nu[\vec{x} := \vec{d}] \models b \triangleright \top \text{ implies } T_D(A), \nu[\vec{x} := \vec{d}] \models t^* \preceq u. \quad (4.6)$$

**Proof.** Let $t = \sum_{\vec{x}} t^* \preceq b \triangleright \delta$, and let $\nu_\ell$ be the interpretation homomorphism from $\text{Pol}(A, D)$ into $T_D(A)$ generated by $\nu$; then

$$\nu_\ell(t) = \sum\{\nu_\ell(t^*[\vec{x} := \vec{d}]) | \vec{d} = d_1, \ldots, d_n \in D \text{ s.t. } \nu(b[\vec{x} := \vec{d}]) = \top \}.$$
If $T_D(A).\nu \models t \preceq u$, then, by (GA1),
\[ \iota_\nu(t^*[\bar{x} := \bar{d}]) \leq \iota_\nu(u) \] for all sequences $\bar{d}$ such that $D.\nu[\bar{x} := \bar{d}] \models b \approx \top$.

Since $x_i \not\in \text{FV}(u)$ for all $1 \leq i \leq n$,
\[ \iota_\nu(u) = \iota_\nu(u[\bar{x} := \bar{d}]). \]

Hence $T_D(A).\nu[\bar{x} := \bar{d}] \models t^* \preceq u$.

Conversely, suppose
\[ D.\nu[\bar{x} := \bar{d}] \models b \approx \top \] implies $T_D(A).\nu[\bar{x} := \bar{d}] \models t^* \preceq u$ for all $\bar{d}$.

By (GA2),
\[ T_D(A).\nu[\bar{x} := \bar{d}] \models t \preceq u. \]

Hence, since $\iota_\nu(u) = \iota_\nu(u[\bar{x} := \bar{d}]), T_D(A).\nu \models t \preceq u$. \hfill \square

So, if $t = t_1 + \ldots + t_n$ and $t_i$ is simple for all $1 \leq i \leq n$, then, by (4.4) and (4.5), whether a statement of the form $T_D(A).\nu \models t \preceq u$ is true is determined by whether statements of the form $T_D(A),\nu \models t_i \preceq u$ are true. Furthermore, if $t_i = \sum x_i t_i^* \prec b \triangleright \delta$, then, by (4.6), whether the statement $T_D(A),\nu \models t_i \preceq u$ is true is determined by whether a statement of the form $T_D(A),\nu \models t_i^* \preceq u$ is true. Note that if $t_i$ is simple, then $t_i^*$ is either an action expression or a sequential composition that starts with an action expression.

Let us fix an action expression $a$ and a tree form $t'$, and suppose that $t^* = a$ or $t^* = at'$. We shall now analyse when $\nu$ satisfies $t^* \preceq u$ in $T_D(A)$; again we distinguish cases, this time according to the form of $u$:

Suppose that $u = \delta$; then, by Lemma 2.7(i),

if $t^* = a$ or $t^* = at'$, then $T_D(A),\nu \not\models t^* \preceq \delta$. \hfill (4.7)

Suppose that $u = u' + u''$; then, by Lemma 2.7(ii),

if $t^* = a$ or $t^* = at'$, then
\[ T_D(A),\nu \models t^* \preceq u' + u'' \] if, and only if,
\[ T_D(A),\nu \models t^* \preceq u' \text{ or } T_D(A),\nu \models t^* \preceq u'. \] \hfill (4.8)

For the case that $u$ is a simple expression, we first prove a lemma.

**Lemma 4.6** Suppose $t^* = a$ or $t^* = at'$, and let $\bar{x} = x_1, \ldots, x_n$ be a sequence of variables such that $\{\bar{x}\} \cap \text{FV}(t^*) = \emptyset$; then
\[ T_D(A),\nu \models t^* \preceq \sum x_i u^* \prec b \triangleright \delta \] if, and only if,
\[ \text{there is a sequence } \bar{d} = d_1, \ldots, d_n \in D \text{ such that } \]
\[ D,\nu[\bar{x} := \bar{d}] \models b \approx \top \text{ and } T_D(A),\nu[\bar{x} := \bar{d}] \models t^* \preceq u^*. \] \hfill (4.9)
Proof. Let \( u = \sum_{x} u^* < b \triangleright \delta \), and let \( \iota_{\nu} \) be the interpretation homomorphism from \( \text{Pol}(A, D) \) into \( T_{D}(A) \) generated by \( \nu \); then

\[
\iota_{\nu}(u) = \sum_{\nu(x) = \bar{d}} \iota_{\nu}(u^*[\bar{x} := \bar{d}]) \mid \bar{d} = d_1, \ldots, d_n \in D \text{ s.t. } D, \nu[\bar{x} := \bar{d}] \models b \approx \top.
\]

Since \( \iota_{\nu}(t^*) = \iota_{\nu}(a) \) or \( \iota_{\nu}(t^*) = \iota_{\nu}(a) \cdot \iota_{\nu}(t') \) and \( \iota_{\nu}(a) \) is a tree action, we find by Lemma 2.7(iii) that \( \iota_{\nu}(t^*) \leq \iota_{\nu}(u) \) if, and only if, there exists \( \bar{d} = d_1, \ldots, d_n \in D \) such that \( D, \nu[\bar{x} := \bar{d}] \models b \approx \top \) and \( \iota_{\nu}(t^*) \leq \iota_{\nu}(u^*[\bar{x} := \bar{d}]) \); the lemma follows. \( \square \)

Now, suppose that \( u \) is a simple expression, say \( u = \sum_{x} u^* < b \triangleright \delta \) with \( u^* = a' \) or \( u^* = a' a' \); we conclude our analysis by distinguishing cases according to the forms of \( t^* \) and \( u^* \):

if \( t^* = a \) and \( u^* = a' \), then, by Lemma 2.7(v),

\[
T_{D}(A), \nu \models t^* \ll u^* \text{ if, and only if, } T_{D}(A), \nu \models t^* \approx u^*.
\]

if \( t^* = a t' \) and \( u^* = a' u' \), then, by Lemma 2.7(vi),

\[
T_{D}(A), \nu \models t^* \ll u^* \text{ if, and only if, } T_{D}(A), \nu \models a \approx a', \ t^* \approx u'.
\]

if \( t^* = a t' \) and \( u^* = a' \), or \( t^* = a \) and \( u^* = a' u' \), then, by Lemma 2.7(iv),

\[
T_{D}(A), \nu \not\models t^* \ll u^*.
\]

Our analysis shows that a statement \( T_{D}(A), \nu \models t \ll u \) is equivalent to a first-order combination of statements of the form

1. \( D, \nu \models b \approx \top \), with \( b \) a Boolean expression;
2. \( T_{D}(A), \nu \models a \approx a' \), where \( a \) and \( a' \) are action expressions; and
3. \( T_{D}(A), \nu \models t' \ll u' \) and \( T_{D}(A), \nu \models u' \ll t' \), where \( t' \) and \( u' \) are continuations of simple expressions in \( t \) and \( u \), respectively.

It is straightforward to associate an appropriate first-order formula with a statement of the first form: conceive \( b \) as an open first-order formula (cf. Proposition 4.4 and the remarks directly following its proof).

Definition 4.7 Suppose that \( D \) has equality; we associate with every two action expressions \( a = a(d_1, \ldots, d_m) \) and \( a' = a'(e_1, \ldots, e_n) \) a Boolean expression \( \text{eq}(a, a') \) as follows:

\[
\text{eq}(a, a') = \begin{cases} 
\text{eq}(d_1, e_1) \land \cdots \land \text{eq}(d_m, e_n) & \text{if } a = a' \text{ and } m = n; \text{ and} \\
\bot & \text{otherwise}
\end{cases}
\]

If we take \( \text{eq}(a, a') \) as a first-order formula, then we have the following lemma.

Lemma 4.8 If \( D \) has equality, then

\[
T_{D}(A), \nu \models a \approx a' \text{ if, and only if, } D, \nu \models \text{eq}(a, a').
\]
Proof. We have

\[ T_D(A), \nu \models a \approx a' \]
\[ \iff a(\bar{\nu}(d_1), \ldots, \bar{\nu}(d_m)) = a'(\bar{\nu}(e_1), \ldots, \bar{\nu}(e_n)) \]
\[ \iff a = a', \ m = n \ \text{and} \ \bar{\nu}(d_i) = \bar{\nu}(e_i) \ \text{for all} \ 1 \leq i \leq n \]
\[ \iff a = a', \ m = n \ \text{and} \ D, \nu \models \text{eq}(d_i, e_i) \ \text{for all} \ 1 \leq i \leq n \]
\[ \iff a = a', \ m = n \ \text{and} \ D, \nu \models \text{eq}(d_1, e_1) \land \cdots \land \text{eq}(d_n, e_n) \]
\[ \iff D, \nu \models \text{eq}(a, a'), \]
by which the lemma is proved.  \( \square \)

Thus, we associate with a statement of the second form the first-order formula \( \text{eq}(a, a') \). With statements of the third form we are going to deal recursively. First, we associate with every tree form \( t \) a natural number \( |t| \):

\[
|\delta| = 0; \quad |\sum_{x} a \otimes b \triangleright \delta| = 1; \\
|t' + t''| = |t'| + |t''|; \quad |\sum_{x} a t' \otimes b \triangleright \delta| = |t'| + 1.
\]

If \( t' \) is the continuation of a simple expression in \( t \), then \( |t'| < |t| \). Consequently, if \( t' \) and \( u' \) are continuations of simple expressions in \( t \) and \( u \), respectively, then

\[ |t'| + |u'| < |t| + |u|. \]

Hence, by induction on \( |t| + |u| \) it follows that the expression \( T_D(A), \nu \models t \ll u \) is equivalent to a first-order combination of expressions of the first two forms. The recursive algorithm in Table 4.1 reflects our analysis, except that it applies (4.6) and (4.9) without verifying the provisos of Lemmas 4.5 and 4.6. Let us say that the algorithm in Table 4.1 is correct for \( t \) and \( u \) if for every variable \( x \)

(i) if \( \sum_{x} \) occurs in \( t \), then \( x \) does not occur at all in \( u \); and

(ii) if \( \sum_{x} \) occurs in \( u \), then \( x \) does not occur at all in \( t \).

Proposition 4.9 Suppose that \( D \) has equality. If the algorithm in Table 4.1 is correct for \( t \) and \( u \), then it associates with \( t \) and \( u \) a first-order formula \( \phi_{=} (t, u) \) such that

\[ T_D(A), \nu \models t \ll u \] if, and only if, \( D, \nu \models \phi_{=} (t, u) \).

Proof. The proof is by induction on \( |t| + |u| \). If \( |t| + |u| = 0 \), then \( t = \delta \), so \( T_D(A), \nu \models t \ll u \) by (4.4) and \( D, \nu \models \phi_{=} (t, u) \) since \( \phi_{=} (t, u) = \top \). Suppose that \( |t| + |u| > 0 \); we proceed by distinguishing cases according to the form of \( t \). We shall only treat the cases that involve an application of the induction hypothesis.

First, suppose that \( t = \sum_{x} a t' \otimes b \triangleright \delta \). Since the algorithm is correct for \( t \) and \( u \), \{\( \bar{\bar{f}}_i \} \cap \text{FV}(a t') = \emptyset \) for all \( 1 \leq i \leq m \), so by (4.9) and (4.11)

\[ T_D(A), \nu \models a t' \ll u \] if, and only if, there exists a sequence \( \bar{d} \) such that

\[ D, \nu[\bar{\bar{f}}_i := \bar{d}] \models b_i \approx \top, \] and \[ T_D(A), \nu[\bar{\bar{f}}_i := \bar{d}] \models a \approx a_i, \ t' \approx u_i. \]
4.2 The definition of $\phi$

compute $\phi_\prec(t, u)$:

let $u = u_1 + \cdots + u_m + u_{m+1} + \cdots + u_n$,

where $u_i = \left\{ \begin{array}{ll}
\sum_{x_i} a_i \cdot u'_i < b_i \triangleright \delta & 1 \leq i \leq m; \\
\sum_{x_i} a_i < b_i \triangleright \delta & m < i \leq n.
\end{array} \right.$

case

$t = \delta$:
return $\top$.

t = \sum_{x} a \triangleleft b \triangleright \delta$:
return

$$\left( \forall \overline{x} \right) \left( b \rightarrow \bigvee_{m < i \leq n} (\exists x_i) (b_i \land \text{eq}(a, a_i)) \right).$$

$t = \sum_{x} a \cdot t' \triangleleft b \triangleright \delta$:
compute $\phi_\prec(t', u'_i)$ for all $1 \leq i \leq m$;
compute $\phi_\prec(u'_i, t')$ for all $1 \leq i \leq m$;
return

$$\left( \forall \overline{x} \right) \left( b \rightarrow \bigvee_{1 \leq i \leq m} (\exists x_i) (b_i \land \text{eq}(a, a_i) \land \phi_\prec(t', u'_i) \land \phi_\prec(u'_i, t')) \right).$$

$t = t' + t''$:
compute $\phi_\prec(t', u)$;
compute $\phi_\prec(t'', u)$;
return $\phi_\prec(t', u) \land \phi_\prec(t'', u)$.

end.

Table 4.1: The algorithm that computes $\phi_\prec$. 

By Lemma 4.8
\[ T_D(A), \nu[\bar{x}_i := \bar{d}] \models a \approx a_i \text{ if, and only if, } D, \nu[\bar{x}_i := \bar{d}] \models \text{eq}(a, a_i), \]
so with two applications of the induction hypothesis, using that \( T_D(A), \nu \models p \approx q \)
if, and only if, \( T_D(A), \nu \models p \preceq q \) and \( T_D(A), \nu \models q \preceq p \), we get
\[ T_D(A), \nu[\bar{x}_i := \bar{d}] \models t' \approx u'_i \text{ if, and only if, } D, \nu[\bar{x}_i := \bar{d}] \models \phi_\approx(t', u'_1) \land \phi_\approx(u'_1, t'). \]
Hence
\[ T_D(A), \nu \models at' \preceq u_i \text{ if, and only if, } D, \nu \models (\exists \bar{x}_i)(b_i \land \text{eq}(a, a_i) \land \phi_\approx(t', u'_i) \land \phi_\approx(u'_i, t')). \]
Consequently, by (4.8) and (4.12)
\[ T_D(A), \nu \models at' \preceq u \text{ if, and only if, } D, \nu \models \bigvee_{1 \leq i \leq m} (\exists \bar{x}_i)(b_i \land \text{eq}(a, a_i) \land \phi_\approx(t', u'_i) \land \phi_\approx(u'_i, t')). \]
Also since the algorithm is correct for \( t \) and \( u, \{\bar{x}\} \cap \text{FV}(u) = \emptyset \), so by (4.6)
\[ T_D(A), \nu \models \sum_\bar{x} at' < b \Rightarrow \exists \delta \ll u \text{ if, and only if, } D, \nu \models \bigvee_{m < i \leq n} (\exists \bar{x}_i)(b_i \land \text{eq}(a, a_i) \land \phi_\approx(t', u'_i) \land \phi_\approx(u'_i, t')) \text{ for all sequences } \bar{d} \text{ such that } D, \nu[\bar{x} := \bar{d}] \models b \approx t. \]
Hence \( T_D(A), \nu \models t \ll u \text{ if, and only if, } D, \nu \models \phi_\ll(t, u) \).
Next, suppose that \( t = t' + t'' \); by the induction hypothesis
\[ T_D(A), \nu \models t' \ll u \text{ if, and only if, } D, \nu \models \phi_\ll(t', u), \]
and
\[ T_D(A), \nu \models t'' \ll u \text{ if, and only if, } D, \nu \models \phi_\ll(t'', u), \]
so by (4.5), \( T_D(A), \nu \models t \ll u \text{ if, and only if, } D, \nu \models \phi_\ll(t, u) \).

The algorithm in Table 4.1 yields a partial recursive function \( \phi_\ll : T_o \times T_o \to \Phi \)
that is defined on \( t \) and \( u \) if the algorithm is correct for \( t \) and \( u \). It induces a total function on \( \alpha \)-congruence classes of tree forms which is by Convention 3.7
also denoted by \( \phi_\ll; \) we have that
\[ T_D(A), \nu \models t \ll u \text{ if, and only if, } D, \nu \models \phi_\ll(t, u). \]
Since \( T_D(A), \nu \models p \approx q \text{ if, and only if, } T_D(A), \nu \models p \preceq q \) and \( T_D(A), \nu \models q \preceq p \),
and by Corollary 3.23, we get that
\[ T_D(A), \nu \models p \approx q \text{ if, and only if, } D, \nu \models \phi_\ll(\theta_o(p), \theta_o(q)) \land \phi_\ll(\theta_o(q), \theta_o(p)). \]
Thus, we have a candidate for $\phi$, except that it is not one-one. (If $t$ is an ordered tree form, then $\theta_o(t + \delta) = t$, so $\phi_\theta(\theta_o(t), \theta_o(q)) = \phi_\theta(\theta_o(t + \delta), \theta_o(q))$ for all $q$.) We obtain a one-one function as follows. Let $\iota : \mathcal{P} \to (\omega - \{0\})$ be any recursive injection of $\mathcal{P}$ into the set of positive natural numbers (any recursive coding of strings over the set of symbols used to write pCRL expressions will do; it is well-known that such codings exist for finite strings over a countable alphabet). For $n \geq 1$ we define $(\bot)^n$ by $(\bot)^1 = \bot$ and $(\bot)^{n+1} = (\bot)^n \lor \bot$: note that $D, \nu \models x \lor \bot^n$ if, and only if, $D, \nu \models x$, for all formulas $x$. Now, let $\phi : \mathcal{P} \times \mathcal{P} \to \Phi$ be such that for all $p$ and $q$

\[
\langle p, q \rangle \mapsto (\phi_\theta(\theta_o(p), \theta_o(q)) \land \phi_\theta(\theta_o(q), \theta_o(p))) \lor (\bot)^r x \lor (\bot)^r y
\]

Then, $\phi$ is the one-one recursive function we needed to define; we have proved

**Theorem 4.10** Suppose that $D$ has equality. Then there exists a one-one recursive function $\phi : \mathcal{P} \times \mathcal{P} \to \Phi$ such that for all pCRL expressions $p$ and $q$

\[
T_D(A), \nu \models p \approx q \text{ if, and only if, } D, \nu \models \phi(p, q).
\]

### 4.3 The definition of $\eta$

We shall now associate with every first-order formula $x$ a pair of pCRL expressions $\eta(x) = \langle p, q \rangle$ such that $D, \nu \models x \lor \bot$ if, and only if $T_D(A), \nu \models p \approx q$. Recall that an open first-order formula may be viewed as a Boolean expression (cf. Section 4.1).

**Lemma 4.11** If $x$ is an open first-order formula and $c$ is a closed action expression, then

\[
T_D(A), \nu \models c \ll x \gg \delta \approx c \text{ if, and only if, } D, \nu \models x.
\]

**Proof.** By Proposition 4.2, $D, \nu \models x \lor \bot$ if, and only if, $D, \nu \models x \lor \bot$ if, and only if, $D, \nu \models x \lor \bot$. If $D, \nu \models x$, then $\iota_\nu(c \ll x \gg \delta) = \iota_\nu(c)$; otherwise $\iota_\nu(c \ll x \gg \delta) = \delta$. Since $\iota_\nu(c) \neq \delta$ the lemma follows.

A formula $x$ is in *prenex form* if it has the form

\[
(Qx_1) \ldots (Qx_n)y
\]

where each $(Qx_i)$ is either $(\exists x_i)$ or $(\forall x_i)$, the variables $x_1, \ldots, x_n$ are all distinct, and $y$ is open. We call $(Qx_1) \ldots (Qx_n)$ the *prefix* of $x$ and $y$ the *matrix*.

**Lemma 4.12** There exists a recursive function $\pi : \Phi \to \Phi$ that associates with every first-order formula $x$ a prenex form $\pi(x)$ such that

\[
D, \nu \models \pi(x) \text{ if, and only if, } D, \nu \models x.
\]

Lemma 4.11 shows how an open first-order formula can be expressed as a pCRL equation. We shall prove now that universal and existential quantifiers can be expressed as transformations on pairs of pCRL expressions. Then, we shall conclude that every prenex form is expressible as a pCRL equation, and we shall define the function $\eta$ using $\pi$ (with a similar trick as in the definition of $\phi$ to ensure that $\eta$ is one-one).

Since universal quantification generalises conjunction, it is instructive to see how conjunction is expressible.

**Example 4.13** Suppose that $t_1$, $t_2$, $u_1$ and $u_2$ are trees. We wish to construct trees $t$ and $u$ such that $t = u$ if, and only if, $t_1 = u_1$ and $t_2 = u_2$. Let $a_1$ and $a_2$ be distinct tree actions; we define

$$t = a_1 \cdot t_1 + a_2 \cdot t_2$$

and

$$u = a_1 \cdot u_1 + a_2 \cdot u_2$$

(see Figure 4.1).

![Figure 4.1: $t = u$ if, and only if, $t_1 = u_1$ and $t_2 = u_2$.](image)

By Lemma 2.7(vi) $a_1 \cdot t_1 = a_1 \cdot u_1$ if, and only if, $t_1 = u_1$, and also, since $a_1 \neq a_2$, $a_1 \cdot t_1 \neq a_2 \cdot u_2$. Hence by Lemma 2.7(ii) $a_1 \cdot t_1 \leq u$ if, and only if, $t_1 = u_1$. Similarly it follows that $a_2 \cdot t_2 \leq u$ if, and only if, $t_2 = u_2$, so $t \leq u$ if, and only if, $t_1 = u_1$ and $t_2 = u_2$. By a symmetric argument it also follows that $u \leq t$ if, and only if, $t_1 = u_1$ and $t_2 = u_2$; we get $t = u$ if, and only if, $t_1 = u_1$ and $t_2 = u_2$.

Let $a$ be a unary parametrised action symbol; we define

$$\forall x_1(p, q) = \sum_x a(x)p; \text{ and}$$

$$\forall x_2(p, q) = \sum_x a(x)q.$$ 

Intuitively, $a(x)$ pairs a particular instance of $p$ with the same instance of $q$: if $d_1, d_2 \in D$ are distinct, then it is possible that $\iota_\nu(p[x := d_1]) = \iota_\nu(q[x := d_2])$ for some valuation $\nu$, while $\iota_\nu(a(d_1)) \neq \iota_\nu(a(d_2))$ implies that

$$\iota_\nu(a(d_1)) \cdot \iota_\nu(p[x := d_1]) \neq \iota_\nu(a(d_2)) \cdot \iota_\nu(q[x := d_2]).$$

Compare this to the use of $a_1$ and $a_2$ in Figure 4.1: it follows from $a_1 \neq a_2$ that $a_1 \cdot t_1 \neq a_2 \cdot u_2$. 


4.3 The definition of $\eta$

**Lemma 4.14** ($\forall$-introduction) If $p$ and $q$ are pCRL expressions, then

$$T_D(A), \nu \models (\forall x)_1(p, q) \approx (\forall x)_2(p, q)$$

if, and only if,

$$T_D(A), \nu[x := d] \models p \approx q \text{ for all } d \in D.$$

**Proof.**

($\Rightarrow$) If $T_D(A), \nu \models (\forall x)_1(p, q) \approx (\forall x)_2(p, q)$, then

$$\sum \{ \iota_\nu(a(d_1)) \cdot \iota_\nu(p[x := d_1]) \mid d_1 \in D \} =$$

$$\sum \{ \iota_\nu(a(d_2)) \cdot \iota_\nu(q[x := d_2]) \mid d_2 \in D \},$$

so by Lemma 2.7(iii,vi), for every $d_1 \in D$ there exists $d_2 \in D$ such that $a(d_1) = a(d_2)$ and $\iota_\nu(p[x := d_1]) = \iota_\nu(q[x := d_2])$. Since $a(d_1) = a(d_2)$ implies $d_1 = d_2$, it follows that

$$\iota_\nu(p[x := d]) = \iota_\nu(q[x := d]) \text{ for all } d \in D;$$

hence $T_D(A), \nu[x := d] \models p \approx q$.

($\Leftarrow$) If $T_D(A), \nu[x := d] \models p \approx q$ for all $d \in D$, then

$$\iota_\nu(a(d)) \cdot \iota_\nu(p[x := d]) = \iota_\nu(a(d)) \cdot \iota_\nu(q[x := d]),$$

so $T_D(A), \nu \models (\forall x)_1(p, q) \approx (\forall x)_2(p, q)$. 

Existential quantification generalises disjunction; the following example explains how disjunction is expressible.

**Example 4.15** Suppose that $t_1$, $t_2$, $u_1$ and $u_2$ are trees. We wish to construct trees $t$ and $u$ such that $t = u$ if, and only if, $t_1 = u_1$ or $t_2 = u_2$. Let $a_1$, $a_2$ and $c$ be distinct tree actions; we define $t = c \cdot (a_1 \cdot t_1 + a_2 \cdot u_2) + c \cdot (a_1 \cdot u_1 + a_2 \cdot t_2)$ and $u = c \cdot (a_1 \cdot t_1 + a_2 \cdot u_2) + c \cdot (a_1 \cdot u_1 + a_2 \cdot t_2) + c \cdot (a_1 \cdot t_1 + a_2 \cdot t_2)$ (see Figure 4.2).

Clearly, $t \leq u$ and $c \cdot (a_1 \cdot t_1 + a_2 \cdot u_2) + c \cdot (a_1 \cdot u_1 + a_2 \cdot t_2) \leq t$; so $t = u$ if, and only if, $c \cdot (a_1 \cdot t_1 + a_2 \cdot u_2) + c \cdot (a_1 \cdot u_1 + a_2 \cdot t_2) \leq t$. Hence, by Lemma 2.7(ii,vi), $t = u$ if, and only if, $t_1 = u_1$ or $t_2 = u_2$.

Let $c$ be a closed action expression and let $a$ be a unary parametrised action symbol; we define

$$(\exists x)_1(p, q) = \sum_x c \left( \sum_x a(x)p + a(x)q \right); \text{ and}$$

$$(\exists x)_2(p, q) = (\exists x)_1(p, q) + c \left( \sum_x a(x)p \right).$$

Note that in the definition of $(\exists x)_1(p, q)$ the first (i.e., left-most) occurrence of $\sum_x$ binds the variable $x$ in $a(x)q$, while the second occurrence binds the variable $x$ in $a(x)p$. Intuitively, by executing $c$ an instance $a(d) \cdot q[x := d]$ of $a(x)q$ is fixed, but from the execution of $c$ it cannot be seen which particular element of $D$ is selected. Compare this to the function of the tree action $c$ in Figure 4.2: by executing $c$ a choice is made between $a_i \cdot t_i$ and $a_i \cdot u_i$ for $i = 1, 2$. 
Lemma 4.16 (\(\exists\)-introduction) If \(p\) and \(q\) are pCRL expressions, then

\[
T_D(A), \nu \models (\exists x)_1 \angle p, q \angle \approx (\exists x)_2 \angle p, q \angle \text{ if, and only if,} \\
\text{there exists } d \in D \text{ such that } T_D(A), \nu[x := d] \models p \approx q.
\]

Proof. Note that

\[
T_D(A), \nu \models (\exists x)_1 \angle p, q \angle \approx (\exists x)_2 \angle p, q \angle \\
\iff T_D(A), \nu \models c(c(\sum_x a(x)p) \leq c(\sum_x a(x)p + a(x)q)) \\
\iff \text{there exists } d \in D \text{ such that} \\
T_D(A), \nu[x := d] \models c(c(\sum_x a(x)p) \leq c(\sum_x a(x)p + a(x)q)) \\
\iff \text{there exists } d \in D \text{ such that} T_D(A), \nu[x := d] \models a(x)q \leq \sum_x a(x)p
\]

and, since \(a(d_1) = a(d_2)\) if, and only if, \(d_1 = d_2\),

\[
\iff \text{there exists } d \in D \text{ such that} T_D(A), \nu[x := d] \models p \approx q.
\]

\[\square\]

Theorem 4.17 There exists a one-one recursive function \(\eta : \Phi \to P \times P\) such that for every first-order formula \(\varphi\)

\[
D, \nu \models \varphi \text{ if, and only if, } T_D(A), \nu \models \varphi \approx \varphi,\text{ where } \eta(\varphi) = \langle p, q \rangle
\]

(provided there are at least a closed action expression and a parametrised action symbol with arity > 0).

Proof. Let \(\varphi\) be a prenex form; we define pCRL expressions \(P(\varphi)\) and \(Q(\varphi)\) as follows:
4.4 A universal fragment

1. if the prefix of \( \varphi \) is empty, i.e., \( \varphi \) is an open formula, then \( P(\varphi) = c < \varphi > \delta \) and \( Q(\varphi) = c \), where \( c \) is a closed action expression;

2. if the prefix of \( \varphi \) begins with a universal quantifier, say \( \varphi = (\forall x)\psi \), then
   \[
P(\varphi) = (\forall x)_1(P(\psi), Q(\psi)) \text{ and } Q(\varphi) = (\forall x)_2(P(\psi), Q(\psi));
   \]
   and

3. if the prefix of \( \varphi \) begins with an existential quantifier, say \( \varphi = (\exists x)\psi \), then
   \[
P(\varphi) = (\exists x)_1(P(\psi), Q(\psi)) \text{ and } Q(\varphi) = (\exists x)_2(P(\psi), Q(\psi)).
   \]

By Lemmas 4.11, 4.14 and 4.16 and an easy induction on the length of the prefix of \( \varphi \) it follows that

\[
D, \nu \models \varphi \text{ if, and only if, } T_D(A), \nu \models P(\varphi) \approx Q(\varphi).
\]

To ensure that \( \eta \) is one-one, we use a recursive injection \( \gamma : \Phi \rightarrow (\omega - \{0\}) \) of \( \Phi \) into the set of positive natural numbers; we define the function \( \eta : \Phi \rightarrow \mathcal{P} \times \mathcal{P} \) by

\[
\varphi \mapsto (P(\pi(\varphi)) + (\delta)^{\varphi^{-1}}, Q(\pi(\varphi))),
\]

where \( (\delta)^1 = \delta \) and \( (\delta)^{n+1} = \delta \cdot (\delta)^n \) for \( n \geq 1 \).

Clearly, \( \eta \) satisfies the requirements of the theorem, so the proof is complete. \( \square \)

By Theorem 4.10 the pCRL theory of \( D \) is one-one reducible to the first-order theory of \( D \), and Theorem 4.17 proves the converse. Hence, the pCRL theory and the first-order theory of \( D \) have the same degree of unsolvability with respect to one-one reducibility. By a theorem of Myhill (see Rogers, Jr., 1992) we get the following corollary.

**Corollary 4.18** If \( D \) has equality, then the pCRL theory of \( D \) and the first-order theory of \( D \) are recursively isomorphic (provided there are at least a closed action expression and a parametrised action symbol with arity > 0).

### 4.4 A universal fragment

The choice quantifier is a powerful construct: it may be used to simulate both the universal and the existential quantifier of first-order logic. Indeed, the algorithm of Table 4.1 yields an open formula when applied to tree forms \( t \) and \( u \) without choice quantifiers, and with any open formula Lemma 4.11 associates a pCRL expression without choice quantifiers. The main application of choice quantifiers is to model input. We shall now investigate how much of the expressiveness of choice quantifiers persists if we only use it to model input.

In Section 3.6 we have introduced a fragment of value-passing CCS. We have associated with every process expression of that language a pCRL expression. Thus, value-passing CCS gives rise to a fragment of pCRL: a pCRL expression that is associated with some process expression of value-passing CCS we have called an
input/output expression. The input/output theory of $D$ consists of all pCRL equations $p \approx q$, with $p$ and $q$ input/output expressions, such that $T_D(A) \models p \approx q$. We shall see below that the input/output theory of $D$ is essentially less complex than the full pCRL theory of $D$: it is recursively isomorphic to the universal fragment of the first-order theory of $D$. We easily get a variant of Lemma 4.11.

**Lemma 4.19** Suppose that $\varphi$ is an open first-order formula, and let $c$ be a closed output action. Then $T_D(A), \nu \models (\varphi \rightarrow c) \approx c$ if and only if, $D, \nu \models \varphi$.

If $p$ and $q$ are input/output expressions, then $(\forall x)_1(p, q)$ and $(\forall x)_2(p, q)$ are also input/output expressions:

$$(\forall x)_1(p, q) = a?x.p; \text{ and}$$

$$(\forall x)_2(p, q) = a?x.q.$$  

Hence, we have the following lemma.

**Lemma 4.20** (\forall-introduction) If $p$ and $q$ are input/output expressions, then

$$T_D(A), \nu \models (\forall x)_1(p, q) \approx (\forall x)_2(p, q) \text{ if and only if,}$$

$$T_D(A), \nu[x := d] \models p \approx q \text{ for all } d \in D.$$  

A first-order formula is universal if it is in prenex form and all quantifiers in its prefix are universal; we denote by $\Phi_U$ the set of universal formulas. From Lemmas 4.19 and 4.20 we straightforwardly get a variant of Theorem 4.17.

**Theorem 4.21** There exists a one-one recursive function $\eta_\circ : \Phi_U \rightarrow IO \times IO$ such that for every universal first-order formula $\varphi$

$$D, \nu \models \varphi \text{ if and only if, } T_D(A), \nu \models p \approx q, \text{ where } \eta_\circ(\varphi) = (p, q)$$

(provided there is a closed output action and a parametrised action symbol with arity $> 0$).

The transformation $((\exists x)_1, (\exists x)_2)$ defined in Section 4.3 uses a distinct feature of the choice quantifier that is not expressible by means of an input prefix: the variable $x$, bound by the left-most choice quantifier in

$$\sum_x c(\sum_x a(x)p + a(x)q)$$

does not occur in the action expression $c$ that immediately follows it. Recall that, intuitively, by executing $c$ an instance $a(d) \cdot q[x := d]$ of $a(x)q$ is fixed, but from the execution of $c$ it cannot be seen which particular element of $D$ is selected.

From Lemma 3.26 on p. 48 we get that if $p$ is an input/output expression, then the ordered tree form $\theta_\circ(p)$ associated to $p$ has explicit instantiation. We shall now prove that all existential quantifiers can be eliminated from the formula $\sigma_\circ(t, u)$ if $t$ and $u$ are ordered tree forms with explicit instantiation.
Theorem 4.22 Suppose that D has equality, and let t and u be ordered tree forms with explicit instantiation. Then there exists a universal first-order formula \( \varphi \) such that \( D \models \phi_\varphi(t, u) \iff \varphi \).

Proof. We shall apply a few elementary results of first-order logic that are proved, e.g., by Shoenfield (1967); in particular we need the following results on quantifiers:

\[
\begin{align*}
((\forall x)\varphi \land \psi) & \iff (\forall x)((\varphi \land \psi) ; (4.13) \\
((\forall x)\varphi \lor \psi) & \iff (\forall x)((\varphi \lor \psi) ; (4.14) \\
(\varphi \rightarrow (\forall x)\psi) & \iff (\forall x)(\varphi \rightarrow \psi) ; (4.15) \\
(\exists x)(\text{eq}(x, d) \land \varphi) & \iff \varphi[x := d] .
\end{align*}
\]

The proof is by induction on \(|t| + |u|\); we shall only do the induction step. Suppose \(|t| + |u| > 0\); we distinguish cases according to the form of t.

If \( t = \delta \), then \( \phi_\varphi(t, u) = T \), which is a universal formula.

If \( t = t' + t'' \), then by the induction hypothesis \( \phi_\varphi(t', u) \) and \( \phi_\varphi(t'', u) \) are equivalent to universal first-order formulas, say \( (\forall x_1)\ldots(\forall x_k)\varphi' \) and \( (\forall y_1)\ldots(\forall y_l)\varphi'' \). Without loss of generality we may assume that \( x_i \neq y_j, x_i \not\in \text{FV}(\varphi'') \) and \( y_j \not\in \text{FV}(\varphi') \), for all \( 1 \leq i \leq k \) and \( 1 \leq j \leq l \). Hence by (4.13)

\[
\phi_\varphi(t, u) = \phi_\varphi(t', u) \land \phi_\varphi(t'', u) \\
\iff (\forall x_1)\ldots(\forall x_k)\varphi' \land (\forall y_1)\ldots(\forall y_l)\varphi'' \\
\iff (\forall x_1)\ldots(\forall x_k)(\forall y_1)\ldots(\forall y_l)(\varphi' \land \varphi'').
\]

In the two cases that remain t is a simple expression; we shall only treat the case that t has a continuation. Suppose \( t = \sum a_i \cdot t' < b \bowtie \delta \) and let \( u = u_1 + \cdots + u_m + u_{m+1} + \cdots + u_n \) with

\[
u_i = \left\{ \begin{array}{ll}
\sum a_i \cdot u_i' < b_i \bowtie \delta & 1 \leq i \leq m; \\
\sum a_i < b_i \bowtie \delta & m < i \leq n.
\end{array} \right.
\]

Then

\[
\phi_\varphi(t, u) = (\forall \vec{x}) \left(b \rightarrow \bigvee_{1 \leq i \leq m} (\exists \vec{x}_i)(b_i \land \text{eq}(a, a_i) \land \phi_\varphi(t', u_i') \land \phi_\varphi(u_i', t')) \right).
\]

Now consider the subformula

\[
(\exists \vec{x}_i)(b_i \land \text{eq}(a, a_i) \land \phi_\varphi(t', u_i') \land \phi_\varphi(u_i', t')).
\]

By (4.14) and (4.15) it suffices to prove that it is equivalent to a universal formula. By the induction hypothesis \( \phi_\varphi(t', u_i') \) and \( \phi_\varphi(u_i', t') \) are equivalent to universal formulas, say \( (\forall x_1)\ldots(\forall x_k)\varphi \) and \( (\forall y_1)\ldots(\forall y_l)\psi \). If \( |\vec{x}_i| = 0 \), then the theorem follows immediately from (4.13), and if \( \text{eq}(a, a_i) = \bot \), then the theorem follows since \( (\exists x)\bot \iff \bot \). Otherwise \( a \) and \( a_i \) are instances of the same parametrised
action symbol and, since \( u \) has explicit instantiation, \( a_i = a(\vec{x}_i) \). Let \( a = a(\vec{d}) \), where \( \vec{d} \) is a sequence of data expressions with \( |\vec{x}_i| = |\vec{d}| \). Then,

\[
eq(a, a_i) = \eq(x_{i1}, d_1) \land \cdots \land \eq(x_{ik}, d_k).
\]

whence by (4.16)

\[
(\exists \vec{x}_i)(b_i \land \eq(a, a_i) \land \varphi \land \psi) \leftrightarrow (b_i[\vec{x}_i := \vec{d}] \land \varphi[\vec{x}_i := \vec{d}] \land \psi[\vec{x}_i := \vec{d}]).
\]

From this the theorem follows, since by (4.13) the right-hand side is equivalent to a universal formula.

Hence, the universal fragment of the first-order theory of \( D \) is one-one reducible to the input-output theory of \( D \), and from Lemma 3.26 and Theorem 4.22 we get the converse. Hence, the input/output theory of \( D \) and the universal fragment of the first-order theory of \( D \) have the same degree of unsolvability with respect to one-one reducibility. Consequently, by a theorem of Myhill (see Rogers, Jr., 1992) we get the following

**Corollary 4.23** If \( D \) has equality, then the input/output theory of \( D \) and the universal fragment of the first-order theory of \( D \) are recursively isomorphic (provided there exist a closed output action and a parametrised action symbol with arity \( > 0 \)).

**Bibliographic notes**

Ponsé (1996) investigated the complexity of another fragment of \( \mu\text{CRL} \). He considers data algebras with recursive functions and relations, and, with respect to our fragment, he omits the choice quantifiers and includes data-parametric recursion. For (pairs of) specifications in this fragment he classifies a number of properties in the Arithmetical Hierarchy. In particular, he shows that, restricting to computable data, equivalence between two recursive specifications in his fragment is complete in \( \Pi_1^0 \). So, approximately, the contribution of data-parametric recursion to \( \mu\text{CRL} \) corresponds to the contribution of universal quantifiers to first-order logic.

Hennessy and Lin (1995) have already proved part of Corollary 4.23 for value-passing CCS, giving an algorithm that associates to each pair of finite value-passing processes a universal formula that holds if, and only if, the processes are bisimilar. Theorem 4.21 extends their result with the converse, that the universal quantifiers introduced by their algorithm cannot be eliminated.

There is a vast literature exploring the connection between process theory and modal logic (see Bradfield and Stirling (2001) and Stirling (2001) for recent accounts). The connection proceeds via labeled transition systems: a process can be viewed as a labeled transition system modulo bisimulation, a modal formula can be viewed as the specification of a property of a state in a labeled transition system. Incidentally, a labeled transition system may be conceived as a first-order model, interpreting the transition relation as a family of binary relations indexed by the labels. This point of view gives rise to a correspondence between process theory
and first-order logic quite different from the one considered in this chapter. In this context, Hollenberg (1998) studies which operations on labeled transition systems are \textit{first-order definable}, i.e., definable through a set of first-order formulas.