Formal Specification F28FS2, Lecture 9
Relation operations; operation schema composition

Jamie Gabbay

March 5, 2014

Remember

- A relation is a set of maplets.
- A (partial) function is (partial) functional relation.

Remember:

$f : S \nrightarrow T = \mathbb{P}(S \times T)$ maps each $s : S$ to at most one thing on the right.

$f : S \rightarrow T$ maps each $s : S$ to precisely one thing on the right.

$f(s)$ (function application to an element). $R(U)$ (relational image of a set of elements).

If $S' \subseteq S$ and $T' \subseteq T$ then we have

- $S' \lhd f$ and $S' \trianglelefteq f$ (domain restriction and anti-restriction) and
- $f \rhd T'$ and $f \trianglerighteq T'$ (range restriction and anti-restriction).

Sequences

$seq\ T \subseteq \mathbb{N}_1 \nrightarrow T$. $seq\ T$ is sequences; finite lists of elements in T. Know the predicate which characterises the elements of $seq\ T$.

Suppose $L : seq\ T$. Then $head(L) : T$ (first element) $tail(L) : seq\ T$ (rest of the list).

There is also $rev\ L$ (reverse L), $L \oplus L'$ (overwrite L with L'), $L \frown L'$ (concatenate L and L').

There is also $seq_1\ L$ (nonempty sequences) and $iseq\ L$ (injective sequences).

Even more funky things to do with sequences

Suppose $L : seq\ T$.

$last(L) : T$ returns the last element of L. If L is empty $last(L)$ is undefined.

For example $last((tom, dick, harry)) = harry : T$.

$front(L) : seq\ T$ returns all but the last element of L. If L has fewer than two elements, $front(L)$ is undefined.

For example $front((tom, dick, harry)) = (tom, dick) : seq\ T$.

Filtering and squashing

Suppose $L : seq\ T$ and suppose $T' \subseteq T$ (note: equivalently we can suppose $T' : \mathbb{P}\, T$).

Then $L \upharpoonright T'$ is the sequence of elements in L that are also in T'.

For example $\langle tom, dick, harry \rangle \upharpoonright \{tom, harry, jones\} = \langle tom, harry \rangle$.

If $f : \mathbb{N}_1 \twoheadrightarrow T$ is defined on finitely many elements, then $squash(f) : seq\ T$ is the sequence which returns the list of those elements.

For example $squash(\{2 \mapsto dick, 3 \mapsto tom, 7 \mapsto harry\}) = \{1 \mapsto dick, 2 \mapsto tom, 3 \mapsto harry\}$.

Generic constants

How to define things like *seq*, $\upharpoonright$, *head*, *tail*, and so on?

$$
\begin{array}{|l}
\hline
\textsf{T cat} \\
\frown \; : seq\ T \times seq T \rightarrow seq\ T \\
\hline
\forall s, t : seq\ T \bullet \\
s \frown t = s \cup \{ n \in dom(t) \bullet (n + \# s) \mapsto t(n) \} \\
\hline
\end{array}
$$

Try defining *head*, *tail*, *last*, *front*, *rev*, and so on.

Squashing, defined explicitly in Z, just for fun:

T squash

$$squash : (\mathbb{N} \rightarrow T) \rightarrow seq\ T$$

$$\forall f : \mathbb{N} \rightarrow T \bullet$$
$$\#squash(f) = \#f \wedge$$
$$\forall n : dom(f) \bullet squash(f)(\#(0..n \lhd f)) = f(n)$$

Why is $\#squash(f) = \#f$ in there; what does it do?

Disjointness

Suppose $A_1, \ldots, A_n : \mathbb{P}S$.

$\mathit{disjoint}(A_1, \ldots, A_n)$ is true when

$$\forall\, i, j \in 1 \ldots n \bullet A_i \cap A_j \neq \varnothing \Rightarrow i = j$$

or equivalently (taking the contrapositive)

$$\forall\, i, j \in 1 \ldots n \bullet i \neq j \Rightarrow A_i \cap A_j = \varnothing.$$

In words:

"The elements of $(A_1, \ldots, A_n)$ are pairwise disjoint."

(The contrapositive of $P \Rightarrow Q$ is $\neg Q \Rightarrow \neg P$.
Exercise: using truth-tables verify that these are logically equivalent.)

Partition

If $U : \mathbb{P}S$ then the predicate '$(A_1, \ldots, A_n)$ *partition* U' holds when

$disjoint(A_1, \ldots, A_n)$

and furthermore

$\bigcup(A_1, \ldots, A_n) = U$.

In words

"$(A_1, \ldots, A_n)$ *partition* U is true when A_1 to A_n really do partition U."

For example $(\{1, 2\}, \{5\}, \{3, 4\})$ *partition* $\{1, 2, 3, 4, 5\}$ holds.

Labour-saving: *let*

Suppose we have some long expression — e.g. *primes*

$\{x : \mathbb{N} \mid (x \neq 0 \land \forall\, y, z : \mathbb{Z} \bullet y * z = x \Rightarrow 1 \in \{y, z\}) \bullet x\} : \mathbb{P}\mathbb{N}$

— which we use many times in another expression $BLAH$.

We can write this as $let\ primes = \{...\}\ in\ BLAH$.

You can use this in your schemas, if you like.

Labour-saving: operation schema composition

$$\begin{array}{|l}
\multicolumn{1}{l}{A} \\
\hline
a, a', c! : \mathbb{Z} \\
\hline
a' = a + 42 \\
c! = a' \\
\hline
\end{array}$$

$$\begin{array}{|l}
\multicolumn{1}{l}{B} \\
\hline
a, a', b? : \mathbb{Z} \\
\hline
b? < 10 \\
a' = a + b? \\
\hline
\end{array}$$

Labour-saving: operation schema composition

Then $A; B$ is this:

$$
\begin{array}{|l}
\hline
\textit{A; B} \\
a, c! : \mathbb{Z} \\
a', b? : \mathbb{Z} \\
\hline
\exists\, d : \mathbb{Z} \bullet \\
d = a + 42 \land c! = d \land b? < 10 \land a' = d + b \\
\hline
\end{array}
$$