Schemata of Recursive Functions and Iterative Algorithms

Dominique Cansell (Lessy, EBRP)

1 Description

In [2] we presented the new JRA's instantiation context to define closure, fixpoint (Tarski), well-founded (Noether) and recursion. A new instantiation plugin [4] was developed in the EBRP project [5]. In this paper we describe an instantiation of an eventB development using JRA's instantiation context. We use terminal (as well non-terminal) recursive function and we recall some theorems on closure and recursion. Rodin [6] is used to develop and prove all models describe in this paper.

2 Theorems between Closure an Well-founded Relation

Let r be a relation ($r \in S \leftrightarrow S$) then its transitive closure is defined by a fixpoint and

- if r is well-founded then $closure(r)$ is also well-founded and $\forall x \cdot x \in S \Rightarrow x \notin closure(r^{-1})[\{x\}]$
- if r is well-founded and $r^{-1} \in S \nrightarrow S$ then $finite(closure(r^{-1})[\{x\}])$

3 Well-founded Relation and Fixpoint: Recursion

Recursive functions are defined with a well-founded relation and the fixpoint theorem.
- Let r be a well-founded relation on S: $r \in S \leftrightarrow S$
- Let g be a a function such that: $g \in (S \times (S \nrightarrow T)) \rightarrow T$
- There is a unique total function fr: $fr \in S \rightarrow T$

such that we have: $\forall x \cdot x \in S \;\Rightarrow\; fr(x) = g(x \mapsto r^{-1}[\{x\}] \lhd fr)$

- The value of fr at x depends on its value on the set $r^{-1}[\{x\}]$, $FrSB$ is a function (an operator) which gives the recursive fonction fr: $fr \;=\; FrSB(r \mapsto g)$

Many recursive functions have only one recursive call then $r^{-1}[\{x\}]$ is empty (base case) or a singleton then r^{-1} is a function. In this case we define the function (operator) $FrsB1$ where $FrSB1(r \mapsto f0 \mapsto f) = FrSB(r \mapsto g)$ and $g = \{x, h, b \cdot x \in S \wedge h \in S \nrightarrow B \wedge r^{-1}[\{x\}] \subseteq dom(h) \wedge (x \notin ran(r) \Rightarrow b = f0(x)) \wedge (x \in ran(r) \Rightarrow b = f(x \mapsto h(r^{-1}(x)))) \mid x \mapsto h \mapsto b\}$ in this case we have $\forall x \cdot x \in S \wedge x \notin ran(r) \;\Rightarrow\; fr(x) = f0(x)$ and $\forall x \cdot x \in S \wedge x \in ran(r) \;\Rightarrow\; fr(x) = f(x \mapsto fr(r^{-1}(x)))$

4 Terminal recursion

A function fr is terminal recursive if $fr = FrSB1(r \mapsto f0 \mapsto f)$ and $f(x \mapsto y) = y$ then we have $\forall x \cdot x \in S \wedge x \in ran(r) \;\Rightarrow\; fr(x) = fr(r^{-1}(x))$. The function (operator) $FrsB1Ter$ gives the function : $fr = FrSB1Ter(r \mapsto f0)$.

4.1 An abstract machine

Let fr equals $FrSB1Ter(r \mapsto f0)$ and $x \in S$, a variable R , an event which computes $fr(x)$ in one shot

$$\boxed{\textsf{final} \;=\; \textbf{then}\ R := fr(x)\ \textbf{end}}$$

4.2 A refinement

Let y be a variable initialised to x with the invariant $fr(x) = fr(y)$

$$\boxed{\textsf{final} = \ \textbf{when}\ y \notin ran(r)\ \textbf{then}\ R := f0(y)\ \textbf{end}}$$

$$\boxed{\textsf{progress} = \ \textbf{when}\ y \in ran(r)\ \textbf{then}\ y := r^{-1}(y)\ \textbf{end}}$$

The variant is trivially $closure(r^{-1})[\{y\}]$ then progress cannot take the control forever.

4.3 An algorithm

Using JRA's merging rules [1] we obtain the following algorithm:

$$\boxed{y := x;\ \textbf{while}\ y \in ran(r)\ \textbf{do}\ y := r^{-1}(y)\ \textbf{od};\ R := f0(y)}$$

4.4 An example: gcd with mod

Xavier Leroy uses this gcd example in [3] and explains how well-founded relations are important in order to define recursive function. We can define $gcdmod$ with the following definition:
$gcdmod \;=\; FrSB1Ter(\{a, b \cdot b > 0 \wedge a > b | (b \mapsto a \bmod b) \mapsto (a \mapsto b)\} \mapsto (\lambda x \mapsto y \cdot x > y \wedge y \geq 0 \mid x))$

After proving that the relation is well-founded we got for free: $\forall a \cdot a > 0 \ \Rightarrow gcdmod(a \mapsto 0) = a$ and $\forall a, b \cdot a > b \wedge b > 0 \ \Rightarrow gcdmod(a \mapsto b) = gcdmod(b \mapsto (a \bmod b))$

5 Conclusion

If we correctly instantiate S, r and $f0$ in the corresponding context and if we prove the instantiation PO (r is well-founded and its inverse is a function) the instantiation of the algorithm gives for free the instantiated and correct algorithm.

We have similar schemata for non-terminal recursion (with or without stack) and sorted algorithms.

References

1. J.-R. Abrial. *Modeling in Event-B: System and Software Engineering*. Cambridge University Press, 2010
2. D. Cansell, J.-R. Abrial: *Examples of using the Instantiation Plug-in"*, Rodin Workshop 2021
3. Xavier Leroy. *Well-founded recursion done right*. CoqPL 2024
4. G. Verdier and L. Voisin *Context instantiation plug-in: a new approach to genericity in Rodin.*, Rodin Workshop 2021
5. EBRP *Enhancing EventB and Rodin.* https://irit.fr/EBRP
6. *Rodin Platform.* http://www.event-b.org