G^2-Algebras

Claus Hintermeier*, **Hélène Kirchner**

CRIN-CNRS & INRIA-Lorraine
BP239, F-54506 Vandœuvre-lès-Nancy Cedex, France
E-mail: hinterme@loria.fr, hkirchne@loria.fr

G^2-algebras are polymorphic, order-sorted algebras where sorts are terms in an equational theory. G^2-algebras are still first order and have the classical quotient term algebra as initial model. They extend G-algebras [1] conservatively and are close to a two level, hierarchical fragment of unified algebra [4].

In G^2-algebras, we distinguish between *sorts* (or *types*), used for the description of possible arguments and range of a function, and *combinators*, which are algebraic functions. Domains of functions can be rather complex to describe. E.g. the domain of natural number substraction is the set of all pairs of natural numbers $\langle x, y\rangle$, s.t. $x \geq y$. This is not a regular tree language and therefore, it is not possible to describe this domain with the help of flat, linear function declarations only.

One way to describe this domain is to use term declarations and semantic sorts. However, we want to calculate with the resulting definition of natural number substraction in form of a decorated term rewriting system [1]. We had to realize [2], that term declarations are hard to handle and underly strong restrictions w.r.t. their form, if we want to compute effectively with them. Another solution consists in the use of sort functions. E.g., as shown in the following very simple example, we can define sorts corresponding to natural numbers and intervals on them.

Example 1. Consider the following specification in pseudo-OBJ syntax, where we distinguish between capital and small letters:

```
obj NATS
    kind     Nats
    sorts    zero :       -> Nats
             succ : Nats  -> Nats
    sort-var X :: Nats
    ops      0 :          -> zero
             s :   X      -> succ(X).
```

In the initial algebra of this specification, which is unique up to isomorphism, $succ^n(zero)$ is interpreted as singleton $\{s^n(0)\}$ and $Nats$ is the set of all such singletons. Now intervals on natural numbers can be specified as follows:

```
obj INATS
    kind      INats
    import    NATS
    sorts     leq :     Nats      -> INats
              geq :     Nats      -> INats
              between : Nats Nats -> INats
    sort-var  X,Y :: Nats
    subsorts  X < leq(X) < leq(succ(X))
              X < geq(X) > geq(succ(X))
              X < between(X,X)
              between(succ(X), Y) < between(X, Y)
              between(X, Y) < between(X, succ(Y)).
```

* Visiting Prof. P.D. Mosses at BRICS 09-12/1994, partially supported by ESPRIT Basic Research Working Group COMPASS

Now, substraction maybe defined as follows:

```
obj NAT-SUBSTRACTION
    import    INATS
    sort-vars X::Nats
    vars      x::X, y::leq(X)
    ops       _∸_ :X leq(X) -> geq(zero)
    axioms    x ∸ 0          = x
              s(x) ∸ s(y)    = x ∸ y.
```

The great difference between the last axiom and a conditional many-sorted specification like:

$$y \leq x \Rightarrow s(x) \dot{-} s(y) = x \dot{-} y,$$

is the model notion. G^2-models do not force $\dot{-}$ to be defined over all naturals, as this is the case for models of many-sorted conditional specifications. G^2-algebras allow to specify precisely the domain of a function.

Suppose now that we have an analogous specification module $INTS$ for integers *ints*, using an additional unary minus function $-$. Consequently, there may be two combinator terms representing zero: 0 and -0, a situation very common in real life computing. Addition can then easily be defined as follows:

```
obj INT-ADDITION
    import INTS
    ops    _+_ : ints ints -> ints
    vars   x, y::ints
    axioms x + 0          = x
           x + -0         = x
           x + s(y)       = s(x + y)
           ...
```

Now, addition `plus` may be defined in the standard way on sorts, too, i.e. with only one zero, namely `zero`. A kind of verification of the implementation of addition on the integers results then in proving that `x + y` is of sort `X plus Y` by structural induction.

The presented framework has meanwhile been superseded by an n-level Horn clause approach called G^n-logics [3], having a sound and complete deduction system and initial models. Furthermore, sort terms are still interpreted as sets and can be used to specify function graphs. Another feature of G^n-logics is the possibility to mix combinator and sort terms, such as `create(nat)` for the creation of lists of natural numbers. Finally, let us mention that membership formulas, i.e. typing, can be conditional. Hence, G^n-logics are by far more expressive than G^2-algebras.

References

1. C. Hintermeier, C. Kirchner, and H. Kirchner. Dynamically-typed computations for order-sorted equational presentations —extended abstract. In S. Abiteboul and E. Shamir, editors, *Proc. 21st International Colloquium on Automata, Languages, and Programming*, volume 820 of *Lecture Notes in Computer Science*, pages 450–461. Springer-Verlag, 1994.
2. C. Hintermeier, C. Kirchner, and H. Kirchner. Sort inheritance for order-sorted equational presentations. In *Recent Trends in Data Type Specifications*, Lecture Notes in Computer Science. Springer-Verlag, 1994.
3. C. Hintermeier, H. Kirchner, and P. D. Mosses. R^n- and G^n-logics. Technical report, Centre de Recherche en Informatique de Nancy, 1994.
4. P. D. Mosses. Unified algebras and institutions. In *Proceedings 4th IEEE Symposium on Logic in Computer Science, Pacific Grove*, pages 304–312, 1989.

This article was processed using the LaTeX macro package with LLNCS style