Introduction to Specification in Rewriting Logic

A. Information for lecturers

Unit description

Description: This unit presents the basic concepts of formal specification in rewriting logic, implemented in the programming language Maude. Once the system is specified the students must be able to prove properties on it.
a short description of the unit about its subject matter and organisation, the student level, expected prior knowledge, the significant concepts and essential questions addressed, the course and context in which it has been used in HE practice, and the estimated duration.
Student and discipline level: Master students from computer science engineering.
Prior knowledge: Propositional logic and limited knowledge of declarative programming. No previous knowledge of equational or rewriting logic is expected.
Estimated duration: 10 hours.
Facilities: Personal computer.

Learning objectives

In this unit students will learn how to specify systems. In particular, they will learn:

- To abstract a system. The main requirement when specifying a system is to abstract the particular implementation features to focus on behavior. This abstraction is not trivial, because it must keep the main properties of the system without going into the details.
- To define the underlying data structures (static part of the system) by means of their Mathematical properties in equational logic. The abstract representation of the system forces students to understand whether a data structure behaves as e.g. a set (so it is commutative, associative, and has a unity), a list, etc.
- To define the dynamic behavior of the system by means of rewrite rules, which requires students to understand the possible states a system can reach.
- To prove invariants on these systems. Students need to understand the system, define the appropriate invariants, and check whether they hold in the system.

IBME character of the unit

This unit requires different approaches depending of the contents. The static behavior of the system is taught by presenting different data structures (e.g. stacks, lists, ordered lists, and sets) using interactive demonstrations. The dynamic behavior is first presented using also interactive demonstrations, in this case focusing on games.

Once the main concepts have been introduced, some assignments are handed. The first one follows a guided inquiry approach, but in the second one the inquiry is left open, being in some sense guided via lecturer & student discussions.

Mathematical content

Students will learn about equational logic and rewriting logic.

Technological pedagogical content knowledge

The previous knowledge (four years of computer science engineering) focused on the details of the implementations, so students find great problems when abstracting the system. Even if the instructor presents abstract systems, the students need to work by themselves and face the possible problems to learn it.

Learning path

The unit consists of two parts: the equational (static) part and the rewriting (dynamic) part. Because the dynamic part uses the static one as infrastructure, the transition is very natural. We propose two activities: first, we require a static system to be implemented. Then students are asked to implement a complete system, which requires to define the data structures and implement transitions by means of rewrite rules on them. In this way contents are first presented separately and combined at the end.

Lecturers' experiences in HE classroom practice

Students find great difficulties because this kind of knowledge is very different from the rest of the contents they face. It is important to explain the context and how these elements complement the rest of the Master's course. It is also important to let them fail, so they understand why the system does not work by themselves.

It is worth letting students work in small groups, because it is closer to the real environment they will find and they successfully collaborate to solve each other's problems and doubts.

Some students with special needs may have problems abstracting concepts. In this case it is important to devote more time to the concept.

Assessment activities are attached. As indicated in previous sections, they distinguish between a purely static structure and a dynamic one.

B. Student learning activities

We present two tasks (4 hours each of them). They follow a similar schema: the first two hours are used by the instructor to present the basic ideas using previous knowledge that is now presented in a more formal way. Then, the students have a 2-hours inquiry session where they use the ideas previously discussed.

Task 1

Implement and test different data structures in Maude.

Learning goals: learning how to specify the static behavior of a system using equational logic and how to implement it in Maude.

Significant concepts and essential questions

- How to decide the most appropriate data structure for a system, based on its mathematical properties (such as unordered, ordered, possibly empty, etc).
- How to represent them in equational logic.
- How to implement them in Maude.

Inquiry types and emphasis on *inquiry abilities*

As indicated in the previous section, we first use an interactive demonstration for presenting the main ideas, progressively discussing more complex data structures (all of them linear, such as stacks or lists). Then, in the next session a guided inquiry session is used to discuss how to represent a tree structure. In this second session the students build the complete specification discussing between them (mediated by the instructor). The ideas they propose are really implemented in the same session, so they can test how they work.

Envisioned student engagement / involvement in construction conceptual understanding

The students already know how the discussed data structures are implemented and how they work. However, they do not know how they are abstracted and which are their underlying Mathematical properties. The instructor guides the sessions explaining how this previous knowledge is transformed into the proposed paradigm.

Tool use: whiteboard for discussions, Maude system for implementations.

Time needed: 4 hours (2 hours for the interactive demonstration + 2 hours for the guided inquiry section).

Suggestions for use

Students must know the concepts before starting the session. They also need to think about how to represent the data structure and basic knowledge of Maude syntax. For the assignment, after discussing the main ideas, asking students one by one to give small answers works very well in practice.

Task 2

Implement and analyze the dynamic behavior of systems using games.

Learning goals: learning how to decide the most appropriate data structure for a concurrent system. Learning how to define the transitions of the system by means of rewrite rules.

Significant concepts and essential questions

- Choosing the most appropriate data structure (defined using equational logic) for a given system.
- Representing the transitions of a system using rewrite rules.
- Analyzing systems using searches.

Inquiry types and emphasis on *inquiry abilities*

In the first session some games are presented and solved using rewriting logic. These games are solved by interactive demonstration, discussing with the students the data structures and the strategies required to solve them.

In the second session free inquiry is used to solve another game. The students can discuss with the instructor but they can propose many different solutions and try to implement them to assess whether they ease the implementation or they make it more difficult.

Envisioned student engagement / involvement in construction conceptual understanding

Using games highly motivates students to participate. For this reason, the engagement of students in this task is usually higher than in the previous one. This instructor guides the session but let students to freely discuss among them different proposal

Tool use: whiteboard for discussions, Maude system for implementations.

Time needed: 4 hours (2 hours for discussing different games + 2 hours for implementing a new game).

Suggestions for use

Students are expected to read and think about the problems and Maude syntax beforehand.

C. Worksheet and files

Our unit provides the following folders:

- **assignments**, containing the assignments and divided into:
    - **task1**, containing the assignment for the first task and its solution.
    - **task2**, containing the assignment for the second task and its solution.
- **examples**, containing the examples and divided into:
    - **task1**, containing examples for the first task.
    - **task2,** containing examples for the first task.
- **manual**, containing the manual.