Points to Bear in Mind

B.1 Introduction

- CoFI aims at establishing a wide consensus. ......................... 4
- The focus of CoFI is on algebraic techniques. ......................... 5
- CoFI has already achieved its main aims. ............................. 5
- CoFI is an open, voluntary initiative. ..................................... 6
- CoFI has received funding as an ESPRIT Working Group, and is sponsored by IFIP WG 1.3. ............................................. 6
- New participants are welcome! ............................................. 7
- CASL has been designed as a general-purpose algebraic specification language, subsuming many existing languages. ......................... 7
- CASL is at the center of a family of languages. ....................... 8
- CASL itself has several major parts. .................................... 9

B.2 Underlying Concepts

- CASL is based on standard concepts of algebraic specification. ...... 11
- A basic specification declares symbols, and gives axioms and constraints. ................................................................. 11
- The semantics of a basic specification is a signature and a class of models. ............................................................. 12
- CASL specifications may declare sorts, subsorts, operations, and predicates. ............................................................. 12
- Sorts are interpreted as carrier sets. ........................................ 12
- Subsorts declarations are interpreted as embeddings. ................. 13
- Operations may be declared as total or partial. ....................... 13
- Predicates are different from boolean-valued operations. ............. 13
- Operation symbols and predicate symbols may be overloaded. ...... 14
- Axioms are formulas of first-order logic. ............................... 14
• Sort generation constraints eliminate ‘junk’ from specific carrier sets.

• The semantics of a structured specification is simply a signature and a class of models.

• A translation merely renames symbols.

• Hiding symbols removes parts of models.

• Union of specifications identifies common symbols.

• Extension of specifications identifies common symbols too.

• Free specifications restrict models to being free, with initiality as a special case.

• Generic specifications have parameters, and have to be instantiated when referenced.

• The semantics of an architectural specification reflects its modular structure.

• Architectural specifications involve the notions of persistent function and conservative extension.

• The semantics of a library of specifications is a mapping from the names of the specifications to their semantics.

B.3 Getting Started

• Simple specifications may be written in **Casl** essentially as in many other algebraic specification languages.

• **Casl** provides also useful abbreviations.

• **Casl** allows loose, generated and free specifications.

• **Casl** syntax for declarations and axioms involves familiar notation, and is mostly self-explanatory.

• Specifications can easily be extended by new declarations and axioms.

• In simple cases, an operation (or a predicate) symbol may be declared and its intended interpretation defined at the same time.

• Symbols may be conveniently displayed as usual mathematical symbols by means of **%display** annotations.

• The **%implies** annotation is used to indicate that some axioms are supposedly redundant, being consequences of others.

• Attributes may be used to abbreviate axioms for associativity, commutativity, idempotence, and unit properties.

• Genericity of specifications can be made explicit using parameters.

• References to generic specifications always instantiate the parameters.

• Datatype declarations may be used to abbreviate declarations of sorts and constructors.

• Loose datatype declarations are appropriate when further constructors may be added in extensions.

• Sorts may be specified as generated by their constructors.

• Generated specifications are in general loose.

• Generated specifications need not be loose.