Some technical aspects of an Interlocking Specification Language

Lars-Henrik Eriksson
Industrilogik L4i AB

Industrilogik L4i AB

- Swedish consultancy company in advanced logic, mathematics and computer science.
- Main business area: Formal methods
- Staff of 11. 7 Ph.D.
- Founded in 1997, but most of the staff worked industrially with formal methods before that.

ISL – why yet another language?

The ISL projects aims at developing a specification language tailored for railway signalling.

• We believe that *domain-specific* specification languages can substantially simplify the application of formal methods. (Both from the point of view of the user and of the tool implementor.)

ISL work is still in a preliminary stage – the language is not yet defined.

The ISL is not necessarily intended to be a completely new specification language.

Existing languages (subsets, dialects, libraries) could well be used if they suit the technical requirements.

Aspects of ISL and ISL use

- Specification levels
- Concept library
- Expressiveness
- Temporalities
- Tool support
- Positive requirements

Specification levels

ISL is primarily intended for design-independent requirements specification of signalling systems.

ISL is not intended as a *design* specification language.

Design specification languages (e.g. Dutch EURIS/LARIS, Adtranz Signal’s IRSL) can be tailored for a specific design.

ISL must interface with design specifications.

Concept library

A formal specification can be divided (roughly) into two parts:

1) *The conceptual model*

- defines the concepts needed to express the requirements: geometrical properties and relations, abstract entities like train routes, etc.
- turns out to be large and difficult to write – even for intuitively obvious concepts
- Example: a signal is ”ahead of” another signal. Positions of points etc. must be taken into account.

2) The *actual requirements*

- are generally simple.

ISL should provide a concept library for railway signalling.

Parametrisation of concepts

Different railway administrations use slightly different concepts.

- Example: the exact definition of a train route can differ between two administrations.

The ISL concept library needs to be parametrised and adaptable.

(The ERRI A201 project has developed a catalogue of the differences in signalling principles.)

Expressiveness

Generally, great expressiveness has been a desired property of a specification language.

\+ Specifications become more concise and easier to express.

– A very expressive language is harder to master.

– There is a conflict between expressibility and suitability for processing by automatic tools (e.g. automatic theorem proving).

– Not much expressiveness needed to specifying requirements

\+ Conceptual modelling does benefit.

ISL should provide just the required amount of expressiveness. Possibly different sublanguages could be used for conceptual modelling and requirements specification.

Temporalities

It is essential to be able to express requirements including temporal relationships.

However, temporalities can easily make the language much more complex – particularly from an implementation point of view.

Just what kind of temporalities are really needed?

Tool support

The use of (more or less) automated tools in a formal development process is essential.

ISL should be carefully designed to take allow tools to take maximal advantage of the state-of-the art in algorithms for theorem proving, type checking etc.

The existence of a standardised concept library means that concept definitions could be coded into the tools to improve performance.

Example of tools that should be available:

– Simulation tools
– Verification tools
– Tools to present and edit specifications in an application-oriented manner.

Positive requirements – a challenge

Formal specification work typically address *safety* properties, e.g. that nothing bad happens.

• Example: a signal is not cleared unless it is safe to do so.

Positive requirements – that something good does happen is in practise more difficult to express.

• less clear-cut than safety requirements.

Verifying a safety requirement can be done by showing that each single computation step of the design/implementation does not cause a bad thing to happen.

Verifying a positive requirement can be more difficult as many steps may be required until the good thing eventually happen.

What are the implications for ISL?