

Tool support for CSP-CASL

Andy Gimblett

MPhil project with M. Roggenbach

University of Wales Swansea

BCTCS 2005, Nottingham

Outline

- Motivation: tool support for specification languages
- CSP-CASL
- Frameworks
 - Parsec – a combinator parser in Haskell
 - HETS – the Heterogenous Toolset
- Towards parsing CSP-CASL

Tool Support for
CSP-CASL

Motivation

Tool support for specification languages

- Analysis
 - Parser
 - Static analysis
 - Can have undecidable parts, eg instantiation of parameters in CASL
- Proof support
 - Refinement relations between specifications
 - Consistency
 - Deadlock freedom
- Animation

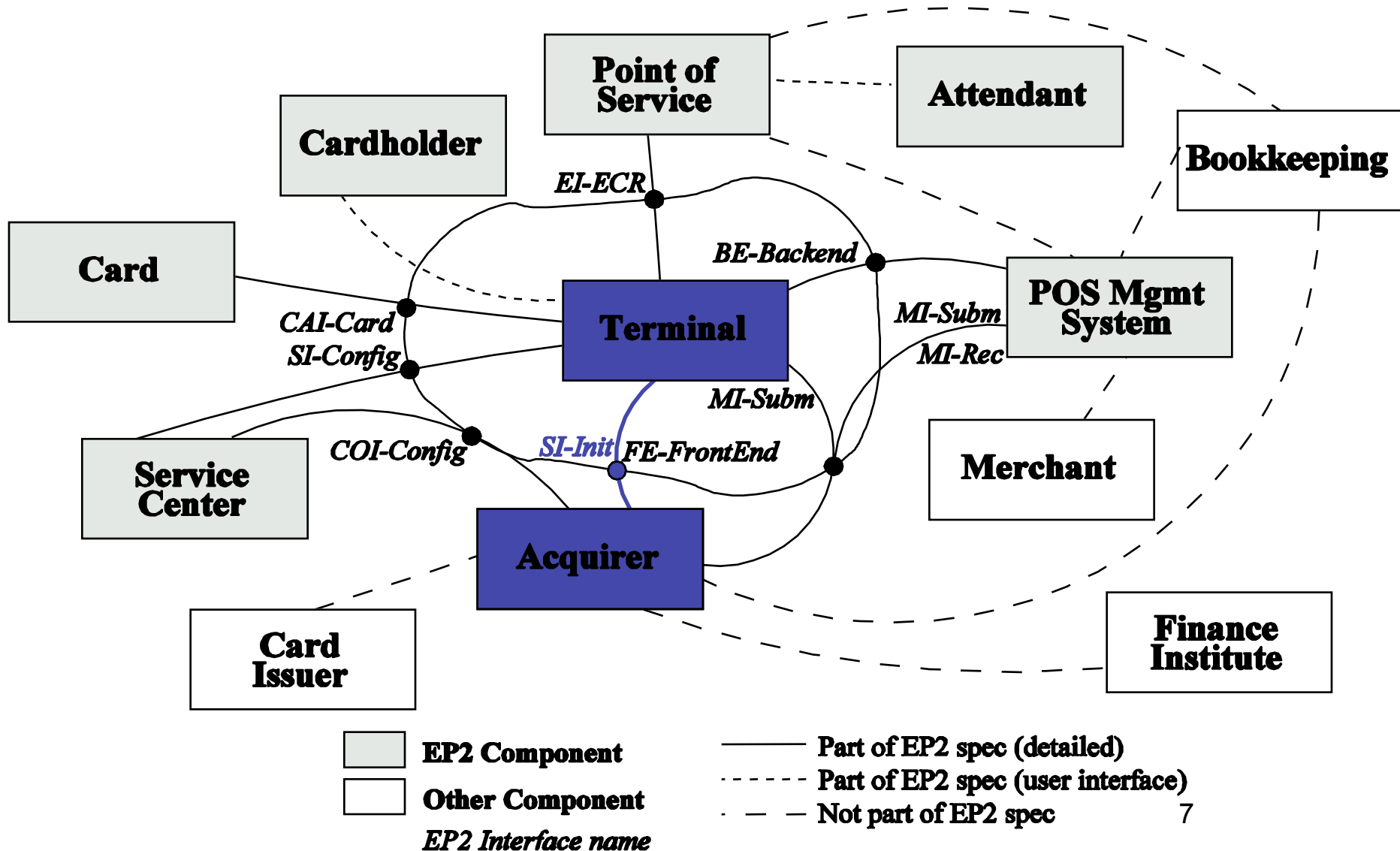
Tool Support for
CSP-CASL

CSP-CASL

CSP-CASL

- CSP
 - Process algebra – reactive systems
 - Roscoe, *The Theory & Practice of Concurrency*, 1998
- CASL
 - Algebraic specification language – abstract data types
 - Mosses (Ed), *CASL Reference Manual*, 2004
- CSP-CASL
 - Development of both data and processes
 - Roggenbach, *CSP-CASL – A New Integration of Process Algebra and Algebraic Specification*, to appear in TCS.

Example: Part of EP2 in CSP-CASL



Example: Part of EP2 in CSP-CASL

ccspec ep2 =

data

sort D_SI_Init;

channel

C_SI_Init: D_SI_Init;

process

let

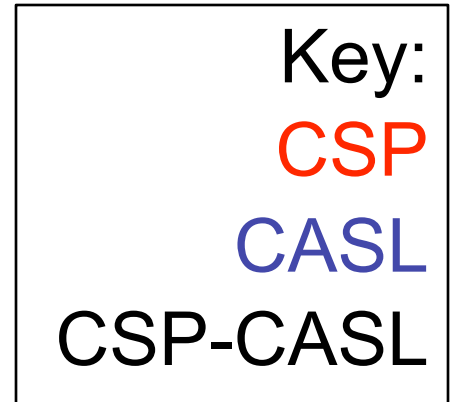
Acquirer = Run(C_SI_Init)

Terminal = Run(C_SI_Init)

in

Terminal [] C_SI_Init [] Acquirer

end



Tool Support for
CSP-CASL

Frameworks

Parsec – combinator parsers in Haskell

- Standard Haskell library
- Top-down, infinite lookahead, context-sensitive
- More direct approach than LL(k)/LR(k)
 - Parsers are first-class values in Haskell
- Parser combinators, eg
 - `<|>` – predictive choice
 - `try` – don't consume input upon failure
 - `many` – accept multiple instances

Parsec – small example

Concrete syntax example:

```
data D_SI_Init
```

Abstract syntax definition:

```
data DATA_DEFN = Data_Defn DATA_NAME [Pos]  
  deriving (Show,Eq)
```

Corresponding parser:

```
dataDefn :: AParser st DATA_DEFN  
dataDefn = do {  
    dataT  
    ; data_name <- simpleID  
    ; return (Data_Defn data_name)  
}
```

HETS – The Heterogeneous Toolset

- Toolset for CASL and CASL extensions (Mainly developed at Bremen University)
 - Parsing/static analysis
 - Interfaces to other tools, eg Isabelle, Maya
- Supported languages:
 - CASL
 - HasCASL
 - **CSP-CASL**
 - ...

Tool Support for
CSP-CASL

**Towards parsing
CSP-CASL**

Tool support for CSP-CASL based on HETS

- New: parsing of processes
- Systematic reuse of HETS
 - Parsing & static analysis of data part
 - Parsing & static analysis of data in the process part
 - CASL structuring mechanisms for CSP-CASL
 - CASL library mechanisms for CSP-CASL

Building a parser

- Top-down approach. Start with:
 ccspec <spec-name> =
 data <casl-part>
 process <csp-part>
 end
- <casl-part>: uses “structured CASL spec” parser/grammar from HETS
- <csp-part>: comes next
- Pretty printer/unparser – in progress
- Everything so far is done “within HETS framework”.

Reuse of HETS

- HETS barely documented – detective work!
- HETS methodology: separate files for keywords, abstract syntax, parsers, ...
 - AS_CSP_CASL.hs, Parse_CSP_CASL.hs, etc.
- Reuse of shared symbols (eg =, end): declarations & parsers from HETS.
- AParser st a – parser with annotations
- Positional information in parse tree
 - data Pos = SourcePos { sourceName :: String,
sourceLine :: !Int, sourceColumn :: !Int }

Example runs

```
> cat ok.csp-casl broken.csp-casl
```

```
ccspec rob = data bucky process satchel end
```

```
ccspec rob = data bucky ferret satchel
```

```
> ./ccparse ok.csp-casl
```

```
Parsed OK.
```

```
Named_csp_casl_spec "rob" (Csp_casl_spec (Data_Defn  
  "bucky") (Process_Defn "satchel"))
```

```
> ./ccparse broken.csp-casl
```

```
broken.csp-casl: parse error at "broken.csp-casl" (line 1,  
  column 25):
```

```
unexpected "f"
```

```
expecting space or "process"
```

Tool Support for
CSP-CASL

**Summary &
Future Work**

Summary

- HETS is complex
- Missing documentation of HETS makes it hard to reuse it
- + Functional programming is suitable for development of efficient (?) parsers
- + Parsec makes parser development intuitive
- + HETS offers a rich infrastructure for parsing any CASL extension

Future Work

- Support full CSP-CASL
- Add pattern matching
- Develop static analysis