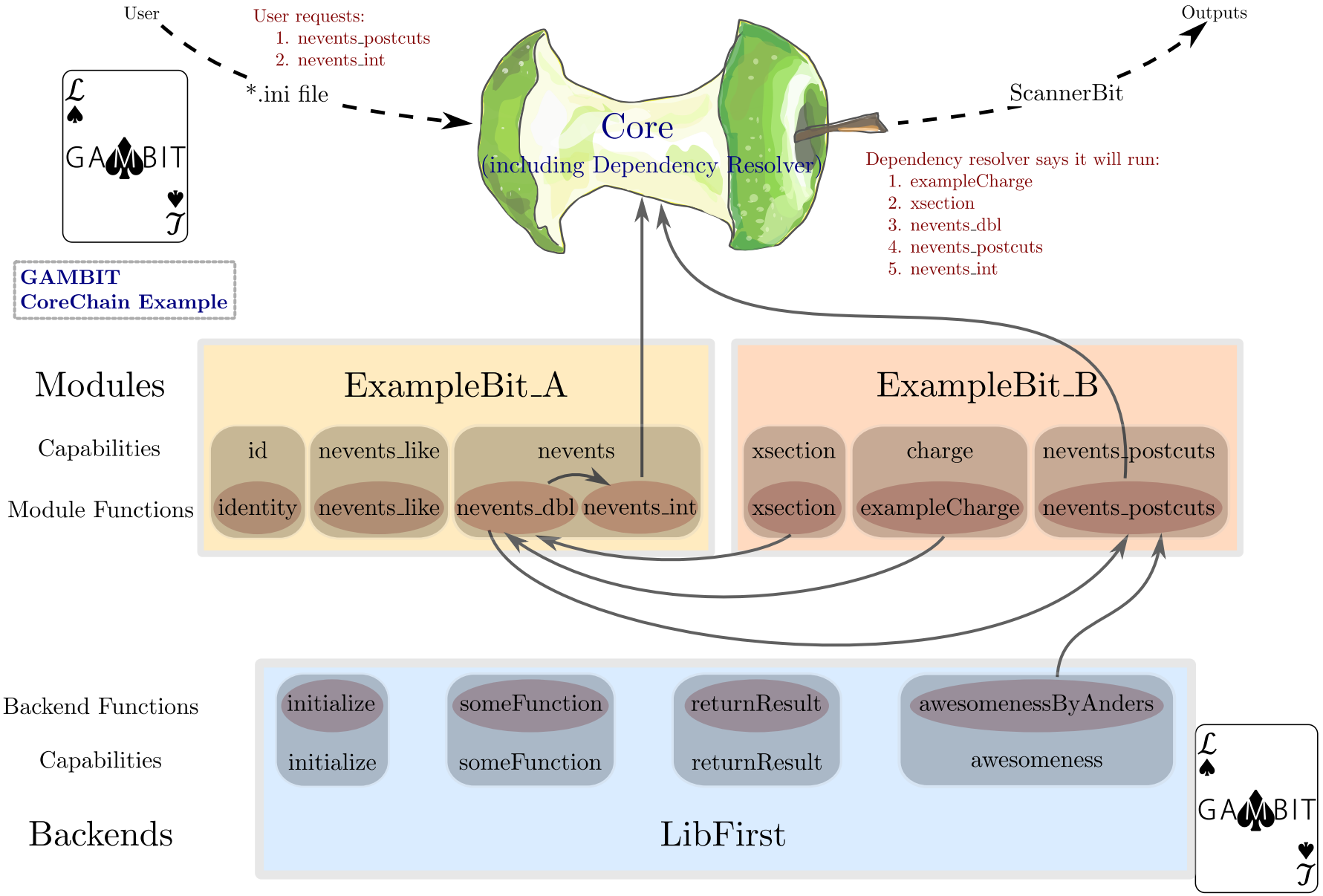
GAMBIT Core + Model News

Pat Scott

McGill/Imperial

Slides available from Workshop Indico page

Status at Gambit II: CoreChain example



Major updates since Stockholm

- Pipes
- Backend extensions
- Parallelised module function evaluation with OpenMP ‘rollcall loops’
- Rollcall extensions
  - Module initialisation functions
  - Conditional backend requirements
  - Error-control in rollcall macros
- Module standalone compilation
- Gambit type-inclusion scheme
- 3 $\times$ main examples
- Errors, logs, names, etc

Pipes

- No more macro calls from within module functions
  $\longrightarrow$ `Pipes::`*function_name*`::x`
- Safe pointers instead to all the GAMBIT things a function might care about. These are the only ways in+out!!
- Here $x =$
  - `Dep::`*dependency_capability* Declared dependencies
  - `BEReq::`*backend_requirement_capability* Declared backend requirements
  - `Models` Current models (only ones this function is allowed to work with)
  - `runOptions` Options passed in from the ini file
  - `Loop::`*stuff* Loop tools (more later)
- Examples:
  - `ExampleBit_A/src/ExampleBit_A.cpp`
  - `ExampleBit_B/src/ExampleBit_B.cpp`

Parallelised module function evaluation with OpenMP 'rollcall loops'

Thread-safe OpenMP parallelised loops over module functions

`ExampleBit_A/include/ExampleBit_A_rollcall.hpp`

`ExampleBit_A/src/ExampleBit_A.cpp`

- `Loop::executeIteration(`*it*`)` Run iteration number *it* of the loop (from Loop Manager)
- `Loop::iteration` Current loop iteration number (in Nested Function)

The following should be obvious, but in case not:

- Any module functions put inside a loop must be threadsafe themselves
- Any calls to backends must be to threadsafe backend functions

Rollcall extensions

- Module initialisation functions

  `ExampleBit_B/src/ExampleBit_B.cpp`
- Conditional backend requirements

  `ExampleBit_B/include/ExampleBit_B_rollcall.hpp`
- Error-control in rollcall macros
- Proposed: compact and/or argument-aware backend req declarations (during the week)

Backend extensions

- Common block / struct access now happens entirely by pointers (BE_VARIABLE)

  `Backends/include/backend_libfirst.hpp`

  `ExampleBit_B/src/ExampleBit_B.cpp`
- Elegant optional array reindexing scheme for Fortran arrays implemented (Farrays; Lars)

  `Backends/lib/libFarrayTest.f90`

  `Backends/include/backend_libFarrayTest.hpp`

  `ExampleBit_A/src/ExampleBit_A.cpp`
- The BOSS is coming. . . (Anders)

Module standalone complilaton

- Fully standalone compilation of GAMBIT physics modules now possible
- Requires inclusion of `Utils/include/standalone.hpp` and *MODULE*`/include/`*MODULE*`_rollcall.hpp` from main file
- All modules must ship with: Models, Backends, Utils, Logs, Printers, contrib
- All modules must ship without: Core, other modules (incl. ScannerBit)
- Type inclusion headers will need to be (re-)written by Python scripts for ease of use

`ExampleBit_A/examples/ExampleBit_A_standalone_example.cpp`

Gambit type-inclusion scheme

- **Common return types:**
  in `Utils/include/shared_types.hpp`
- **Module-specific types:**
  in *MODULE*`/include/`*MODULE*`_types.hpp`, then included in `Utils/include/types_rollcall.hpp`
- **Model-specific types:**
  in `Models/models/`*MODEL*`_types.hpp`, then included in `Utils/include/types_rollcall.hpp`
- **Backend-specific types:**
  in `Backends/include/`*BACKEND*`_types.hpp`, then included in `Utils/include/shared_types.hpp`

Eventually, inclusion of module, model and backend-specific type headers into `shared_types.hpp` and `types_rollcall.hpp` will happen automagically in the harvester script

3 × main examples

- sandbox: `Core/examples/gambit_example.cpp`
- minimal: `Core/examples/gambit_example_minimal.cpp`
- standalone: `ExampleBit_A/examples/ExampleBit_A_standalone_example.cpp`
- Makefile structure currently consists of `makefile.common` called by compiler-specific `makefile.`*my_compiler*

Errors, logs, names, etc

- Naming convention changed GAMBIT→Gambit in all files (`namespace Gambit`)
- Logs, Models, Backends, Printers, Utils, (Priors?)
- Proper exception handling system implemented – Use it!
- Proper logging system implemented – Use it!

`Utils/src/error_handlers.cpp`

`ExampleBit_B/src/ExampleBit_B.cpp`

`gambit.yaml`