

STEP

structures de données

Alain Muller

Télécom SudParis

October 17, 2019

1 Imported domains

```
import entity from "ri.newgen"  
  
import expression from "ri.newgen"  
  
import effect from "effects.newgen"  
  
import statement from "ri.newgen"
```

2 STEP domains

2.1 The `step_directives` resource

The resource `MODULE.step_directives` is produced par the `step_parser` phase and used by the `step_analyse` and `step_compile` phases.

The `map_entity_int` domain is used in `step_bison_parser.c`.

```
map_entity_int = entity->int
```

Domains used in `step_bison_parser.c`, `parser.c`, `directives.c`, `compile.c`

```
step_clause = reduction:map_entity_int + private:entity*  
+ shared:entity* + transformation:int + nowait:unit +  
threadprivate:entity* + copyin:entity* + firstprivate:entity*  
+ schedule:string*
```

Domains used in `step_bison_parser.c`, `parser.c`, `directives.c`, `analyse.c`, `compile.c`

```
step_directive = type : int x persistent block : statement x
clauses : step_clause*
```

```
step_directives = persistent statement->step_directive
```

2.2 The `step_comm` resource

The resource `PROGRAM.step_comm` is initialized by the `step_analyse_init` phase, updated by the `step_analyse` phase and used by the `step_compile` phase.

Domains used in `analyse.c`

```
map_effect_bool = persistent effect -> bool
```

The origins of a SEND (and RECV) region is tracked by the `step_point` and `map_effect_step_point` domains. For a SEND (or RECV) region, the `step_point` associated in the `map_effect_step_point` table gives the previous SEND (or RECV) region before propagation (fields `module`, `stmt` and `data`).

```
step_point = module : entity x persistent stmt : statement x
persistent data : effect
```

```
map_effect_step_point = persistent effect -> step_point
```

Domains used in `analyse.c`, `compile.c`

```
step_comm = path: map_effect_step_point x interlaced:
map_effect_bool x partial: map_effect_bool
```