SaC v1.4

new in v1.4:

- support for non-recursive structs
- support for gpukernel pragmas
- support for tensor comprehensions

1 Program Structure

prg ⇒ *[(module | class)] [interface]* [structdef]* [typedef]* [objectdef]* [function]**

2 Module Declarations

module ⇒ **module** *id* [**deprecated** *str*] **;**

class ⇒ **class** *id* [**deprecated** *str*] **;** *classtype*

classtype ⇒ **classtype** *type* **;**
| **extern classtype ;** *[interface_pragma]**

3 Import / Export

interface ⇒ *(* **import** / **use** *)* *id* **:** *symbolset* **;**
| *(* **export** / **provide** *)* *symbolset* **;**

symbolset ⇒ **all** [**except** { *ext_id* [**,** *ext_id*] }]
| { *ext_id* [**,** *ext_id*] }

4 Structure Definitions

structdef ⇒ **struct** *id* { *[type id [, id]* ;]** } **;**

5 Type Definitions

typedef ⇒ *loctypedef*
| *exttypedef*

loctypedef ⇒ **typedef** *type id* **;**

exttypedef ⇒ **external typedef** *id* **;** *[interface_pragma]**

6 Object Definitions

objectdef ⇒ *(locobjdef | extobjdef)*

locobjdef ⇒ **objdef** *type* *id* = *funcall* ;

extobjdef ⇒ **external objdef** *type* *id* ; *[interface_pragma]**

7 Function Declarations and Definitions

function ⇒ *extfundec [(interface_pragma | funtion_pragma)]**
| *specfundec [function_pragma]**
| *fundef*
| *main*

extfundec ⇒ **external** *varsignature* ;

specfundec ⇒ **specialize** *fixsignature* ;

fundef ⇒ *[(* **inline** | **noinline** *)]* *fixsignature [function_pragma]** *body*

fixsignature ⇒ *fixrets ext_id* (*fixargs*)
| *operator_sig*

varsignature ⇒ *varrets ext_id* (*varargs*)
| *operator_sig*

operator_sig ⇒ *type* (*ext_op*) (*arg*)
| *type* (*ext_op*) (*arg* , *arg*)

fixargs ⇒ *(arg [, arg]* | [***void** *])*

varargs ⇒ *fixargs*
| *arg [, arg]** , ...

arg ⇒ *type [&] id*

fixrets ⇒ *(type [, type]* | [***void** *])*

varrets ⇒ *fixrets*
| *type [, type]** , ...

main ⇒ **int main** (*[***void** *]*) *body*

8 Function Bodies

body ⇒ { [*cachesim_pragma*] [*vardec*]* [*statement*]* [*return*] }

vardec ⇒ *type* *id* [, *id*]* ;

statement ⇒ ;
| *assignment* ;
| *funcall* ;
| *withloop* ;
| *cond*
| *doloop*
| *whileloop*
| *forloop*

return ⇒ **return** [*expr*] ;
| **return** ([*exprs*]) ;

assignment ⇒ *assign_lhs* [, *assign_lhs*]* *assign_op* *expr*
| *assign_lhs* (++ | −)

assign_lhs ⇒ *id*
| *assign_lhs* [*exprs*]
| *assign_lhs* . *id*

assign_op ⇒ (= | += | −= | *= | /= | %=)

cond ⇒ **if** (*expr*) *statementblock* [**else** *statementblock*]

doloop ⇒ **do** *statementblock* **while** (*expr*) ;

whileloop ⇒ **while** (*expr*) *statementblock*

forloop ⇒ **for** (*assignment* [, *assignment*]*
; *expr* ; *assignment* [, *assignment*]*)
statementblock

statementblock ⇒ { [*cachesim_pragma*] [*statement*]* }
| *statement*

9 Expressions

exprs ⇒ *expr* [, *expr*]*

expr_or_dot ⇒ (*expr* | .)

expr_or_mdot ⇒ (*expr* | . | ...)

expr ⇒ *const*
| *qual_ext_id*
| *funcall*
| *withloop*
| *tensor_comp*
| *array*
| *struct*
| *expr* || *expr*
| *expr* && *expr*
| *expr* ? *expr* : *expr*
| (*type*) *expr*
| (*expr*)

arrray ⇒ [[*exprs*]]
| [: *type*]
| *expr* [[*expr_or_mdot* [, *expr_or_mdot*]*]]

struct ⇒ *id* { *exprs* }
| *id* { [. *id* = *expr* [, . *id* = *expr*]*] }
| *expr* . *id*

funcall ⇒ *qual_ext_id* ([*exprs*])
| *unary_prf* (*expr*)
| *qual_ext_op* *expr*
| *binary_prf* (*expr* , *expr*)
| *expr* *qual_ext_op* *expr*
| *ternary_prf* (*expr* , *expr* , *expr*)

tensor_comp ⇒ { *tc_def* [; *tc_def*]* }

tc_def ⇒ *id* -> *expr* [| *tc_constraint*]
| [[*id_or_mdot* [, *id_or_mdot*]*]] -> *expr* [| *tc_constraint*]

tc_constraint ⇒ *expr* (< | <=) (*id* | *id_vec*) [**step** *expr* [**width** *expr*]]
| (*id* | *id_vec*) (< | <=) *expr* [**step** *expr* [**width** *expr*]]
| *expr* (< | <=) (*id* | *id_vec*) (< | <=) *expr*
[**step** *expr* [**width** *expr*]]

10 With-Loops

withloop ⇒ **with** [*generators*] **:** *operations*

generators ⇒ **{** [*withloop_pragma*] [*generator*]* **}**

generator ⇒ (*index_set*) [*generator_pragma*] [**{** [*statement*]* **}**] **:** *gen_exprs* **;**

index_set ⇒ *expr_or_dot* (< | <=) *index_vars* (< | <=) *expr_or_dot*
[**step** *expr* [**width** *expr*]]

index_vars ⇒ *id* [= *id_vec*]
| *id_vec*

id_vec ⇒ **[** [*id* [, *id*]*] **]**

gen_exprs ⇒ **void**
| *expr*
| (*expr* [, *expr*]*)

operations ⇒ **void**
| *operation*
| (*operation* [, *operation*]*)

operation ⇒ **genarray** (*expr* [, *expr*])
| **modarray** (*expr*)
| **fold** ((*qual_ext_id* | *qual_ext_op*) [(*exprs*)] , *expr*)
| **foldfix** ((*qual_ext_id* | *qual_ext_op*) [(*exprs*)] , *expr* , *expr*)
| **propagate** (*id*)

11 Types

type ⇒ *basetype* [*shape_spec*]

shape_spec ⇒ `[ * ]`
| `[ + ]`
| `[` [`.` [`,` `.`]*] `]`
| `[` *nums* `]`

basetype ⇒ *simpletype*
| *usertype*
| *structtype*

simpletype ⇒ **byte**
| **short**
| **int**
| **long**
| **longlong**
| **ubyte**
| **ushort**
| **uint**
| **ulong**
| **ulonglong**
| **float**
| **bool**
| **char**
| **double**

structtype ⇒ [*id* `::`] **struct** *id*

usertype ⇒ [*id* `::`] *id*

12 Identifiers

id_or_mdot ⇒ *(id | . | ...)*

qual_ext_id ⇒ *[id ::] ext_id*

ext_id ⇒ *(id | reservedid)*

reservedid ⇒ **genarray**
| **modarray**
| **fold**
| **foldfix**
| **propagate**
| **all**
| **except**

qual_ext_op ⇒ *[id ::] ext_op*

ext_op ⇒ *(op | reservedop)*

reservedop ⇒ `&`
| `&&`
| `||`
| `!`
| `~`
| `+`
| `-`
| `*`
| `/`
| `%`
| `<=`
| `<`
| `>=`
| `>`
| `»`
| `«`
| `^`
| `++`
| `-`

13 Constants

const	⇒	*numbyte*
	|	*numshort*
	|	*numint*
	|	*numlong*
	|	*numlonglong*
	|	*numubyte*
	|	*numushort*
	|	*numuint*
	|	*numulong*
	|	*numulonglong*
	|	*num*
	|	*float*
	|	*double*
	|	*char*
	|	[*str*]$^{+}$
	|	**true**
	|	**false**
nums	⇒	[*num* [, *num*]*]

14 Builtin Operations

unary_prf ⇒ *(_tob_S_ / _tos_S_ / _toi_S_ / _tol_S_ / _toll_S_)*
| *(_toub_S_ / _tous_S_ / _toui_S_ / _toul_S_ / _toull_S_)*
| **_tof_S_**
| **_tod_S_**
| **_toc_S_**
| **_tobool_S_**
| *(_not_S_ / _not_V_)*
| *(_neg_S_ / _neg_V_)*
| *(_abs_S_ / _abs_V_)*
| **_dim_A_**
| **_shape_A_**

ternary_prf ⇒ **_modarray_AxVxS_**

binary_prf ⇒ *(_add_SxS_ / _add_SxV_ / _add_VxS_ / _add_VxV_)*
| *(_sub_SxS_ / _sub_SxV_ / _sub_VxS_ / _sub_VxV_)*
| *(_mul_SxS_ / _mul_SxV_ / _mul_VxS_ / _mul_VxV_)*
| *(_div_SxS_ / _div_SxV_ / _div_VxS_ / _div_VxV_)*
| *(_mod_SxS_ / _mod_SxV_ / _mod_VxS_ / _mod_VxV_)*
| *(_min_SxS_ / _min_SxV_ / _min_VxS_ / _min_VxV_)*
| *(_max_SxS_ / _max_SxV_ / _max_VxS_ / _max_VxV_)*
| *(_eq_SxS_ / _eq_SxV_ / _eq_VxS_ / _eq_VxV_)*
| *(_neq_SxS_ / _neq_SxV_ / _neq_VxS_ / _neq_VxV_)*
| *(_le_SxS_ / _le_SxV_ / _le_VxS_ / _le_VxV_)*
| *(_lt_SxS_ / _lt_SxV_ / _lt_VxS_ / _lt_VxV_)*
| *(_ge_SxS_ / _ge_SxV_ / _ge_VxS_ / _ge_VxV_)*
| *(_gt_SxS_ / _gt_SxV_ / _gt_VxS_ / _gt_VxV_)*
| *(_and_SxS_ / _and_SxV_ / _and_VxS_ / _and_VxV_)*
| *(_or_SxS_ / _or_SxV_ / _or_VxS_ / _or_VxV_)*
| **_reshape_VxA_**
| **_sel_VxA_**
| **_take_SxV_**
| **_drop_SxV_**
| **_cat_VxV_**

15 Pragmas

interface_pragma ⇒ # **pragma** **linkname** *str*
| # **pragma** **header** *str*
| # **pragma** **linkwith** *[str]*$^{+}$
| # **pragma** **linkobj** *[str]*$^{+}$
| # **pragma** **copyfun** *str*
| # **pragma** **freefun** *str*
| # **pragma** **linksign** [*nums*]
| # **pragma** **refcounting** [*nums*]
| # **pragma** **effect** *qual_ext_id* *[, qual_ext_id]**

withloop_pragma ⇒ # **pragma** **wlcomp** *wc_funcall*
| # **pragma** **nocuda**

generator_pragma ⇒ # **pragma** **gpukernel** **GridBlock** (*num* , *gk_funcall*)

wc_funcall ⇒ **Default**
| **All** ( )
| **Cubes** ( )
| **ConstSegs** (*[* [*nums*] , [*nums*] , *]*$^{+}$ *wc_funcall*)
| **NoBlocking** (*wc_funcall*)
| **BvL0** (*[* [*nums*] , *]*$^{+}$ *wc_funcall*)
| **BvL1** (*[* [*nums*] , *]*$^{+}$ *wc_funcall*)
| **BvL2** (*[* [*nums*] , *]*$^{+}$ *wc_funcall*)
| **Ubv** (*[* [*nums*] , *]*$^{+}$ *wc_funcall*)
| **Scheduling** (*sched_param* , *wc_funcall*)
| **Tasksel** (*tsel_param* , *wc_funcall*)

gk_funcall ⇒ **Gen**
| **ShiftLB** (*gk_funcall*)
| **CompressGrid** ([*nums*] , *gk_funcall*)
| **Permute** ([*nums*] , *gk_funcall*)
| **FoldLast2** (*gk_funcall*)
| **SplitLast** (*num* , *gk_funcall*)
| **PadLast** (*num* , *gk_funcall*)

cachesim_pragma ⇒ # **pragma** **cachesim** *[str]**

function_pragma ⇒ # **pragma** **recountdots**
| # **pragma** **noinline**