NVIDIOSO

1.0

Generated by Doxygen 1.8.7

Wed Jul 30 2014 17:18:20

Contents

Chapter 1

Main Page

NVIDIOSO NVIDIa-based cOnstraint SOlver v. 1.0

```
__CSP/COP REPRESENTATION__
```

VARIABLES:

Variable has variable types.

- bool: true, false
- int: -42, 0, 69
- set of int: {}, {2, 3, 4}, 1..10

We distinguish between four different types of variables, namely:

- FD Variables: standard Finite Domain variables
- SUP Variables: SUPport variable introduced to compute the objective function. These variables have unbounded int domains.
- OBJ Variables: OBJective variables. These variables store the objective value as calculated by the objective function through standard propagation. These variables have unbounded int domains.

DOMAINS:

Domain representation may vary depending on the type of model that is instantiated. In particular, for a CPU model the domains can be represented by lists of sets of domain value. For CUDA models domains are represented as follows. There are two internal representations for an finite domain D depending on whether |D| <= max_vector or not:

- Bitmap: if |D| <= max_vector;
- List of bounds: otherwise.

By default, max_vector is equal to 256. This value can be redefined via and environment variable VECTOR_MAX.

Domains have the following structure:

| EVT | REP | LB | UB | DSZ || ... BIT ... |

where

- EVT: represents the EVenT happened on the domain;
- REP: is the REPresentation currently used; This value can be one of the following:

  - -1, -2, -3, ...: BIT represents a set of 1, 2, 3, ... bitmaps respectively. Each bitmap represents a domain subset of values {LB, UB};
  - 0 : BIT represents a Bitmap of contiguous values starting from LB: LB..VECTOR_MAX.
  - 1, 2, 3, ... : in BIT there are respectively 1, 2, 3, ... pairs of bound. If there are 0 pairs, then there is a unique pair of bounds {LB, UB} in the LB/UB field respectively.
- LB: Lower Bound of the current domain;
- UB: Upper Bound of the current domain;
- DSZ: Domain SiZe where DSZ <= max_vector -> REP = 0. Moreover,
  - {LB, UB}' = {LB, k} {k', UB} -> DSZ' = DSZ - (k' - k + 1);
  - LB' = LB + k -> DSZ' = DSZ - (k - LB + 1);
  - UB' = UB - k -> DSZ' = DSZ - (UB - k + 1);
- BIT: bit vector where
  - REP $<$ 0: there is a total of ($<=$) VECTOR_MAX bits representing REP pairs of bounds. The first part of BIT is used to store REP pairs $<$LB, UB$>$. This bounds do not change anymore even if the correspondend bitmap changes. This is done in order to keep the original offset when clearing bits from the bitmap. The second part of BIT stores the actual bitmaps. Using UB - LB + 1 it is possible to calculate the size of the bitmap and hence the position in BIT of the next pair $<$LB, UB$>$. When REP $<$ 0 the BIT field does not change anymore. The system will use the LB/UB fields to check for the right bitmap in the BIT field.
  - REP = 0: there are UB - LB + 1 <= VECTOR_MAX bits of contiguous domain values starting from 0;
  - REP $>$ 0: each pair of bound is identified as LB, UB (LB = UB if singlet). If REP = 1, then there is only 1 pair of bounds represented by {LB, UB}. If REP $>$ 1, then there are at least 2 pairs in BIT and the LB/UB fields represent respectively the min/max values among all the pairs.

OBSERVATIONS (CUDA implementation):

Shared Memory: 49152 = 48 kB per block -> keep 47 kB available.

- REP $<$ 0 there are 47 $*$ 1024 = 48128 -> (48128 - 5 $*$ 32)/32 = 1499 possible storable values. Worst case: REP = -256 -> 3 $*$ 256 triples = 3 $*$ 256 = 768 $<$ 1499 (-8=256/32).
- REP = 0 and VECTOR_MAX = 4096 the worst case is when there are 4096 sing.: ((4096 + 4096 $*$ 2 $*$ 32) / 8) / 1024 = 32.5 kB $<$ 45 kB ((tot_bits + tot_bits $*$ 2 int $*$ bit_per_int) / B) / kB.
- REP $>$ 0: 45 kB = 11520 int -> 11520 - 5 = 11515 -> 11515/2 (used two int to represent a pair of bounds) = 5757 pairs separated by at least one "hole" from each other -> 5757 $*$ 2 = 11514 such as {0, 1}, {3, 4}, ...

Note

The above observation means that when the domains are greater than 11514 then a check must be performed in order to apply multiple copies from global to share memory if needed.
A domain such as {300, 450} has 150 values $<$ VECTOR_MAX but it still represented as REP $<$ 0. This is done for efficiency reasons, avoiding to store a further base-offset for contiguous domains of size $<$ VECT↩ OR_MAX.
When a domain (or subsets of it) is (are) represented using a bitmap, the values are stored from right to left using "chunks" of 32 bits (considering a 32bit representation for an unsigned int), where the most significan bit is in the leftmost position of the chuck, i.e., it is the 31th bit. For example, the domain {0, 63} is store as |63...32|31...0|. The chunk containing a value val is easily computing by tot_chunks - (val / 32), where tot_chunks is the total number of chunks used for representing a domain. The position of val within the chunk is given by val % 32.

Chapter 2

NVIDIOSO

NVIDIOSO - NVIDIa-based cOnstraint SOlver v. 1.0

Chapter 3

Todo List

Member BoolDomain::get_event () const

implement this function

Member CudaConcreteBitmapList::add (int min, int max)

complete add function to add any bitmap.

Member CudaConcreteDomainBitmap::add (int min, int max)

implement using checks on chunks of bits (i.e. sublinear cost).

Member CudaVariable::set_domain (std::vector< std::vector< int > > elems)

implement set of sets of elements.

Member IntVariable::set_domain (std::vector< std::vector< int > > elems)=0

implement set of sets of elements.

Member SetDomain::get_event () const

implement this function

Chapter 4

Hierarchical Index

4.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 5

Class Index

5.1 Class List

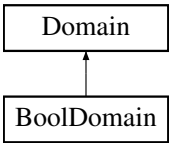
Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 6

Class Documentation

6.1 BoolDomain Class Reference

Inheritance diagram for BoolDomain:



Public Member Functions

- DomainPtr clone () const

  Clone the current domain and returns a pointer to it.
- EventType get_event () const
- size_t get_size () const

  Returns the size of the domain.
- bool is_empty () const

  Returns true if the domain is empty.
- bool is_singleton () const

  Returns true if the domain has only one element.
- void print () const

  Print info about the domain.

Protected Member Functions

- DomainPtr clone_impl () const

  Clone the current domain.

Protected Attributes

- BoolValue _bool_value

  Current domain value.

Additional Inherited Members

6.1.1 Member Function Documentation

6.1.1.1 EventType BoolDomain::get_event ( ) const [virtual]

Get event on this domain

Todo implement this function

Implements Domain.

The documentation for this class was generated from the following files:

- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/bool_domain.h
- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/bool_domain.cpp

6.2 ConcreteDomain< T > Class Template Reference

Public Member Functions

- virtual unsigned int size () const =0
- virtual T lower_bound () const =0

  Returns lower bound.
- virtual T upper_bound () const =0

  Returns upper bound.
- virtual void shrink (T min, T max)=0
- virtual void subtract (T value)=0
- virtual void in_min (T min)=0
- virtual void in_max (T max)=0
- virtual void add (T value)=0
- virtual void add (T min, T max)=0
- virtual bool contains (T value) const =0
- virtual bool is_empty () const =0
- virtual bool is_singleton () const =0
- virtual T get_singleton () const =0
- virtual const void * get_representation () const =0
- virtual void print () const =0

6.2.1 Member Function Documentation

6.2.1.1 template<class T> virtual void ConcreteDomain< T >::add (T *value*) [pure virtual]

It computes union of this domain and {value}.

Parameters

value	it specifies the value which is being added.

Implemented in CudaConcreteBitmapList, CudaConcreteDomainBitmap, and CudaConcreteDomainList.

6.2.1.2 template<class T> virtual void ConcreteDomain< T >::add (T *min*, T *max*) [pure virtual]

It computes union of this domain and {min, max}.

Parameters

min	lower bound of the new domain which is being added.
max	upper bound of the new domain which is being added.

Implemented in CudaConcreteBitmapList, CudaConcreteDomainBitmap, and CudaConcreteDomainList.

6.2.1.3 template<class T> virtual bool ConcreteDomain< T >::contains (T *value*) const [pure virtual]

It checks whether the value belongs to the domain or not.

Parameters

value	to check whether it is in the current domain.

Implemented in CudaConcreteBitmapList, CudaConcreteDomainBitmap, and CudaConcreteDomainList.

6.2.1.4 template<class T> virtual const void* ConcreteDomain< T >::get_representation ( ) const [pure virtual]

It returns a void pointer to an object representing the current representation of the domain (e.g., bitmap).

Returns

void pointer to the concrete domain representation.

Implemented in CudaConcreteDomainBitmap, and CudaConcreteDomainList.

6.2.1.5 template<class T> virtual T ConcreteDomain< T >::get_singleton ( ) const [pure virtual]

It returns the value of type T of the domain if it is a singleton.

Returns

the value of the singleton element.

Note

Classes that specialize this method should handle the case of an invokation of the method and a non-singleton domain. For example, throw an exception or returning the lower bound.

Implemented in CudaConcreteDomainBitmap, and CudaConcreteDomainList.

6.2.1.6 template<class T> virtual void ConcreteDomain< T >::in_max (T *max*) [pure virtual]

It updates the domain according to the maximum value.

Parameters

max	domain value.

Implemented in CudaConcreteBitmapList, CudaConcreteDomainBitmap, and CudaConcreteDomainList.

6.2.1.7 template<class T> virtual void ConcreteDomain< T >::in_min (T *min*) [pure virtual]

It updates the domain according to the minimum value.

Parameters

min	domain value.

Implemented in CudaConcreteBitmapList, CudaConcreteDomainBitmap, and CudaConcreteDomainList.

6.2.1.8 template<class T> virtual bool ConcreteDomain< T >::is_empty ( ) const [pure virtual]

It checks whether the current domain is empty.

Returns

true if the current domain is empty, false otherwise.

Implemented in CudaConcreteDomain.

6.2.1.9 template<class T> virtual bool ConcreteDomain< T >::is_singleton ( ) const [pure virtual]

It checks whether the current domain contains only an element (i.e., it is a singleton).

Returns

true if the current domain is singleton, false otherwise.

Implemented in CudaConcreteDomainBitmap, and CudaConcreteDomainList.

6.2.1.10 template<class T> virtual void ConcreteDomain< T >::print ( ) const [pure virtual]

It prints the current domain representation (its state).

Note

it prints the content of the object given by "get_representation ()" .

Implemented in CudaConcreteDomainBitmap, CudaConcreteBitmapList, and CudaConcreteDomainList.

6.2.1.11 template<class T> virtual void ConcreteDomain< T >::shrink (T *min*, T *max*) [pure virtual]

It updates the domain to have values only within min/max.

Parameters

min	new lower bound to set for the current domain.
max	new upper bound to set for the current domain.

Implemented in CudaConcreteBitmapList, CudaConcreteDomainBitmap, and CudaConcreteDomainList.

6.2.1.12 template<class T> virtual unsigned int ConcreteDomain< T >::size ( ) const [pure virtual]

It returns the number of elements in the domain. It returns the current size of the domain.

Implemented in CudaConcreteBitmapList, CudaConcreteDomainBitmap, and CudaConcreteDomainList.

6.2.1.13 template<class T> virtual void ConcreteDomain< T >::subtract (T *value*) [pure virtual]

It substracts {value} from the current domain.

Parameters

value	the value to subtract from the current domain.

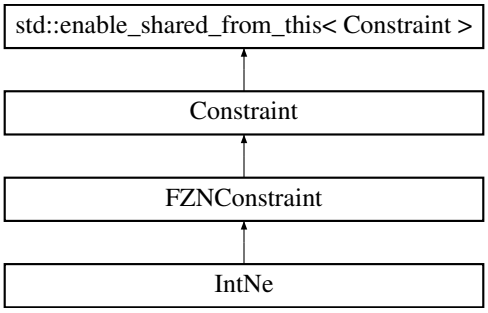
Implemented in CudaConcreteBitmapList, CudaConcreteDomainBitmap, and CudaConcreteDomainList.

The documentation for this class was generated from the following file:

- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/concrete_domain.h

6.3 Constraint Class Reference

Inheritance diagram for Constraint:



Public Member Functions

- size_t get_unique_id () const

  Get unique (global) id of this constraint.
- int get_number_id () const
- std::string get_name () const

  Get the name id of this constraint.
- int get_weight () const

  Get the weight of this constraint.
- void increase_weight (int weight=1)
- void decrease_weight (int weight=1)
- size_t get_scope_size () const
- size_t get_arguments_size () const

  Get the size of the auxiliary arguments of this constraint.
- const std::vector< EventType > & events () const
- const std::vector< int > & arguments () const
- virtual void update (const Event &e)
- virtual std::vector < ConstraintPtr > decompose () const
- virtual std::vector< VariablePtr > changed_vars_from_event (EventType event) const
- virtual std::vector< VariablePtr > changed_vars () const
- virtual bool fix_point () const
- virtual int unsat_level () const
- virtual const std::vector < VariablePtr > scope () const =0
- virtual void attach_me ()=0
- virtual void consistency ()=0
- virtual bool satisfied ()=0
- virtual void remove_constraint ()=0

- virtual void print () const =0
  Prints info.
- virtual void print_semantic () const =0
  Prints the semantic of this constraint.

Protected Member Functions

- Constraint ()
- virtual ConstraintPtr get_this_shared_ptr ()

Protected Attributes

- std::string _dbg
  Debug string.
- int _number_id
- std::string _str_id
- std::vector< EventType > _trigger_events
- std::vector< int > _arguments

6.3.1 Constructor & Destructor Documentation

6.3.1.1 Constraint::Constraint ( ) [protected]

Default constructor. It creates a new instance of a null constraint with a new unique id. It sets all the other members to null.

6.3.2 Member Function Documentation

6.3.2.1 const std::vector< int > & Constraint::arguments ( ) const

It returns the list of auxiliary arguments of a given constraint.

6.3.2.2 virtual void Constraint::attach_me ( ) [pure virtual]

It attaches this constraint (observer) to the list of the variables in its scope. When a variable changes state, this constraint could be automatically notified (depending on the variable).

Implemented in FZNConstraint.

6.3.2.3 std::vector< VariablePtr > Constraint::changed_vars ( ) const [virtual]

It returns the vector of (pointers to) all variables for which the corresponding domains have been modified by the propagation/consistency of this constraint.

Returns

a vector of (pointers to) variables which domains have been modified after the propagation of this constraint. It returns null if no domain has been modified.

6.3.2.4 std::vector< VariablePtr > Constraint::changed_vars_from_event (EventType *event*) const [virtual]

It returns the vector of (pointers to) variables that correspond to the variables for which the domains have been modified by the propagation/consistency of this constraint w.r.t. a given event.

Parameters

event	the event to that may be happened on some domain of the variables of the scope of this constraint.

Returns

a vector of (pointers to) variables which domains have been modified after the propagation of this constraint. It returns null if no domain has been modified.

6.3.2.5 virtual void Constraint::consistency ( ) [pure virtual]

It is a (most probably incomplete) consistency function which removes the values from variable domains. Only values which do not have any support in a solution space are removed.

Implemented in FZNConstraint, and IntNe.

6.3.2.6 std::vector< ConstraintPtr > Constraint::decompose ( ) const [virtual]

It returns a vector of (pointers to) constraints which are used to decompose this constraint. It actually creates a decomposition (possibly also creating variables), but it does not impose the constraints.

Returns

a vector of (pointers to) constraints used to decompose this constraint.

6.3.2.7 void Constraint::decrease_weight (int *weight* = 1)

Decrease current weight.

Parameters

weight	the weight to decrease from the current weight (default: 1).

6.3.2.8 const std::vector< EventType > & Constraint::events ( ) const

It returns the list of events that trigger a given constraint.

6.3.2.9 bool Constraint::fix_point ( ) const [virtual]

It checks if the constraint has reached the fixed point, i.e., it checks whether no events happened on the domains of the variables in the scope of the this constraint.

6.3.2.10 int Constraint::get_number_id ( ) const

Get number id of this constraint.

Note

same type of constraints have same number_id.

6.3.2.11 size_t Constraint::get_scope_size ( ) const

Get the size of the scope of this constraint, i.e., the number of FD variables which is defined on.

Note

The size of the scope does not correspond to the formal definition of the constraint but with the actual number of variables within the scope of a given constraint. For example: int_eq (x, y) has _scope_size equal to 2; int_eq (x, 1) has _scope_size equal to 1.

6.3.2.12 ConstraintPtr Constraint::get_this_shared_ptr ( ) [protected], [virtual]

Create a shared pointer from this instance.

Returns

a shared pointer to Constraint object.

6.3.2.13 void Constraint::increase_weight (int *weight* = 1)

Increse current weight.

Parameters

weight	the weight to add to the current weight (default: 1).

6.3.2.14 virtual void Constraint::remove_constraint ( ) [pure virtual]

It removes the constraint by removing this constraint from all variables in its scope.

Implemented in FZNConstraint.

6.3.2.15 virtual bool Constraint::satisfied ( ) [pure virtual]

It checks if the constraint is satisfied.

Returns

true if the constraint if for certain satisfied, false otherwise.

Note

If this function is incorrectly implementd, a constraint may not be satisfied in a solution.

Implemented in FZNConstraint, and IntNe.

6.3.2.16 virtual const std::vector<VariablePtr> Constraint::scope ( ) const [pure virtual]

It returns the vector of (shared) pointers of all the variables involved in a given constraint (i.e., its scope).

Implemented in IntNe.

6.3.2.17 int Constraint::unsat_level ( ) const [virtual]

It returns an integer value that can be used to represent how much the current constraint is unsatisfied. This function can be used to implement some heuristics for optimization problems.

Returns

an integer value representing how much this constraint is unsatisfied. It returns 0 if this constraint is satisfied.

6.3.2.18 void Constraint::update (const Event & *e*) [virtual]

It receives an update about an action that has been performed on some variables and it acts accordingly. This method is used to trigger some actions when this observer observes a change in the state of some observed subject.

Parameters

e	an object of type Event that specifies the event that triggered the update.

6.3.3 Member Data Documentation

6.3.3.1 std::vector<int> Constraint::_arguments [protected]

It represents the array of auxiliary arguments needed by a given constraint in order to be propagated. For example: int_eq (x, 2) has 2 as auxiliary argument.

6.3.3.2 int Constraint::_number_id [protected]

It specifies the number if for a given constraint. All constraints within the same type have unique number ids.

6.3.3.3 std::string Constraint::_str_id [protected]

It specifies the string id of the constraint. If it is null, then the string id is created from string associated for the constraint type and the _number_id of the constraint.

6.3.3.4 std::vector<EventType> Constraint::_trigger_events [protected]

It specifies the events which trigger the propagation of a given constraint.

Note

see domain.h for the list of events of type "EventType".

The documentation for this class was generated from the following files:

- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/constraint.h
- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/constraint.cpp

6.4 ConstraintStore Class Reference

The documentation for this class was generated from the following files:

- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/constraint_store.h
- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/constraint_store.cpp

6.5 CPModel Class Reference

Public Member Functions

- virtual void add_variable (VariablePtr ptr)
- virtual void add_constraint (ConstraintPtr ptr)
- virtual void add_search_engine (SearchEnginePtr ptr)

Protected Attributes

- std::list< VariablePtr > _variables

  Variables.
- ConstraintPtr _constraint_store

  Constraint Store.
- SearchEnginePtr _search_engine

  Search engine.

6.5.1 Member Function Documentation

6.5.1.1 void CPModel::add_constraint (ConstraintPtr *ptr*) [virtual]

Add a constraint to the model. It linkes constraints to variables, actually defining the constraint graph.

Parameters

ptr	pointer to the constraint to add to the model

6.5.1.2 void CPModel::add_search_engine (SearchEnginePtr *ptr*) [virtual]

Add a search engine to the model.

Parameters

ptr	pointer to the search engine to use to explore the search space.

6.5.1.3 void CPModel::add_variable (VariablePtr *ptr*) [virtual]

Add a variable to the model. It linkes variables to constraints, actually defining the constraint graph.

Parameters

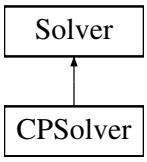
ptr	pointer to the variable to add to the model

The documentation for this class was generated from the following files:

- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/cp_model.h
- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/cp_model.cpp

6.6 CPSolver Class Reference

Inheritance diagram for CPSolver:



Public Member Functions

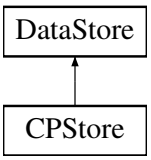
- void **run** ()

The documentation for this class was generated from the following file:

- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/cp_solver.h

6.7 CPStore Class Reference

Inheritance diagram for CPStore:



Public Member Functions

- virtual bool load_model (std::string="")

  Load model from input file (FlatZinc model)
- virtual void init_model ()
- virtual void print_model_info ()

  Print info about the model.
- virtual void **print_model_variable_info** ()
- virtual void **print_model_domain_info** ()
- virtual void **print_model_constraint_info** ()

Static Public Member Functions

- static CPStore * get_store (std::string in_file)

  Constructor get (static) instance.

Protected Member Functions

- CPStore (std::string)

  Protected constructor for singleton pattern.

Additional Inherited Members

6.7.1 Member Function Documentation

6.7.1.1 void CPStore::init_model ( ) [virtual]

Init store with the loaded model. This method works on the internal state of the store. It uses a generator to generate the right instances of the objects (e.g. CUDA-FD variabes) and add them to the model. A generator takes tokens as input and returns the corresponding pointer to the instantiated objects.

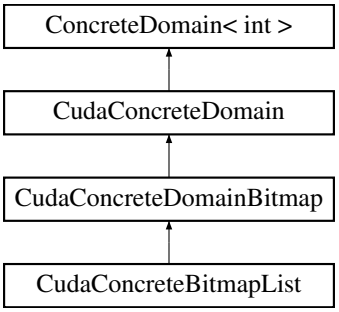
Implements DataStore.

The documentation for this class was generated from the following files:

- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/cp_store.h
- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/cp_store.cpp

6.8 CudaConcreteBitmapList Class Reference

Inheritance diagram for CudaConcreteBitmapList:



Public Member Functions

- CudaConcreteBitmapList (size_t size, std::vector< std::pair< int, int > > pairs)
- unsigned int size () const

  It returns the current size of the domain.
- void shrink (int min, int max)
- void subtract (int value)
- void in_min (int min)
- void in_max (int max)
- void add (int value)
- void add (int min, int max)
- bool contains (int val) const
- void print () const

Protected Member Functions

- int find_pair (int val) const
- int find_prev_pair (int val) const
- int find_next_pair (int val) const

Protected Attributes

- int _num_bitmaps

  Number of pairs in the list (list size).
- int _bitmap_size

  Fixed size of each bitmap in the list.
- unsigned int _domain_size

Additional Inherited Members

6.8.1 Constructor & Destructor Documentation

6.8.1.1 CudaConcreteBitmapList::CudaConcreteBitmapList (size_t *size*, std::vector< std::pair< int, int > > *pairs*)

Constructor. It allocates size bytes for the internal domain's representation and it initializes it with the pairs of bounds contained in pairs.

Parameters

size	the number of bytes to allocate.
pairs	the SORTED list of pairs to allocate.

6.8.2 Member Function Documentation

6.8.2.1 void CudaConcreteBitmapList::add (int *value*) [virtual]

It computes union of this domain and {value}.

Parameters

value	it specifies the value which is being added.

Reimplemented from CudaConcreteDomainBitmap.

6.8.2.2 void CudaConcreteBitmapList::add (int *min*, int *max*) [virtual]

It computes union of this domain and {min, max}.

Parameters

min	lower bound of the new domain which is being added.
max	upper bound of the new domain which is being added.

Note

it is possible to add only bitmaps with empty intersection with previous bitmaps and which min is greater than current lower bound.

Todo complete add function to add any bitmap.

Reimplemented from CudaConcreteDomainBitmap.

6.8.2.3 bool CudaConcreteBitmapList::contains (int *val*) const [virtual]

It checks whether the value belongs to the domain or not.

Parameters

val	to check whether it is in the current domain.

Note

val is given w.r.t. the lower bound of 0.

Reimplemented from CudaConcreteDomainBitmap.

6.8.2.4 int CudaConcreteBitmapList::find_next_pair (int *val*) const [protected]

Find the index of the first pair with values greater than val.

Parameters

val	to be compared in the list of pairs.

Returns

the index of the pair with val greater than val, -1 if no such pair exists.

Note

it returns the index of the pair regardless of whether the element is present or not.

6.8.2.5 int CudaConcreteBitmapList::find_pair (int *val*) const [protected]

Find the index of the pair containing val.

Parameters

val	to be searched in the list of pairs.

Returns

the index of the pair containing val, -1 otherwise.

Note

it returns the index of the pair regardless of whether the element is present or not.

6.8.2.6 int CudaConcreteBitmapList::find_prev_pair (int *val*) const [protected]

Find the index of the last pair with values smaller than val.

Parameters

val	to be compared in the list of pairs.

Returns

the index of the pair with val lower than val, -1 if no such pair exists.

Note

it returns the index of the pair regardless of whether the element is present or not.

6.8.2.7 void CudaConcreteBitmapList::in_max (int *max*) [virtual]

It updates the domain according to max value.

Parameters

max	domain value.

Reimplemented from CudaConcreteDomainBitmap.

6.8.2.8 void CudaConcreteBitmapList::in_min (int *min*) [virtual]

It updates the domain according to min value.

Parameters

min	domain value.

Reimplemented from CudaConcreteDomainBitmap.

6.8.2.9 void CudaConcreteBitmapList::print ( ) const [virtual]

It prints the current domain representation (its state).

Note

it prints the content of the object given by "get_representation ()".

Reimplemented from CudaConcreteDomainBitmap.

6.8.2.10 void CudaConcreteBitmapList::shrink (int *min*, int *max*) [virtual]

It updates the domain to have values only within min/max.

Parameters

min	new lower bound to set for the current domain.
max	new upper bound to set for the current domain.

Reimplemented from CudaConcreteDomainBitmap.

6.8.2.11 void CudaConcreteBitmapList::subtract (int *value*) [virtual]

It substracts {value} from the current domain.

Parameters

value	the value to subtract from the current domain.

Reimplemented from CudaConcreteDomainBitmap.

6.8.3 Member Data Documentation

6.8.3.1 unsigned int CudaConcreteBitmapList::_domain_size [protected]

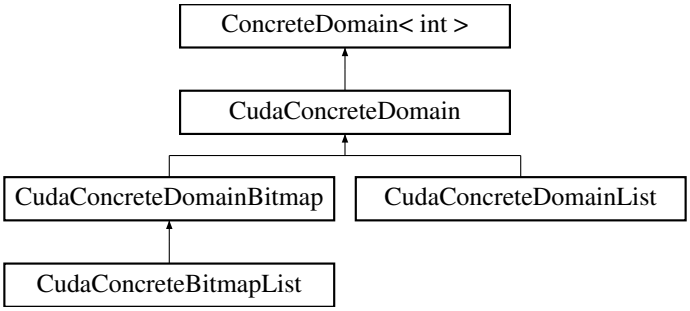
Current domain size, i.e., sum of the elements on each bitmap.

The documentation for this class was generated from the following files:

- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/cuda_concrete_bitmaplist.h
- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/cuda_concrete_bitmaplist.cpp

6.9 CudaConcreteDomain Class Reference

Inheritance diagram for CudaConcreteDomain:



Public Member Functions

- CudaConcreteDomain (size_t size)
- int lower_bound () const

  Returns lower bound.
- int upper_bound () const

  Returns upper bound.
- int get_num_chunks () const
- size_t get_alloc_bytes () const
- bool is_empty () const

Protected Member Functions

- void flush_domain ()
- void set_empty ()

Protected Attributes

- std::string **_dbg**
- int _num_chunks

  Number of allocated (32 bit int) chunks.
- int _lower_bound

  Lower bound.
- int _upper_bound

  Upper bound.
- int * _concrete_domain

6.9.1 Constructor & Destructor Documentation

6.9.1.1 CudaConcreteDomain::CudaConcreteDomain (size_t *size*)

Constructor for CudaConcreteDomain. It instantiates a new object and allocate size bytes for the array of integers

Parameters

size	the number of bytes to allocate.

Note

the client should check whether integers are represented by 32 bit values.

6.9.2 Member Function Documentation

6.9.2.1 void CudaConcreteDomain::flush_domain ( ) [protected]

Flush domain: reduces its domain size to zero by flushing all values in the internal domain's representation. It sets the current domain's state as empty.

Note

it sets upper bound $<$ lower bound.

6.9.2.2 size_t CudaConcreteDomain::get_alloc_bytes ( ) const

Get the number of allocated bytes, i.e., the size of the internal domain's representation.

6.9.2.3 int CudaConcreteDomain::get_num_chunks ( ) const

Get the number of allocated chunks (in terms of 32 bit integers).

6.9.2.4 bool CudaConcreteDomain::is_empty ( ) const [virtual]

It checks whether the current domain is empty.

Returns

true if the current domain is empty, false otherwise.

Implements ConcreteDomain$<$ int $>$.

6.9.2.5 void CudaConcreteDomain::set_empty ( ) [protected]

Empty domain: reduces its domain size to zero by setting the current domain's state as empty.

Note

it does not flush the current internal domain's representation.

6.9.3 Member Data Documentation

6.9.3.1 int* CudaConcreteDomain::_concrete_domain [protected]

Concrete domain is represented by an array of (32 bit) integers.

Note

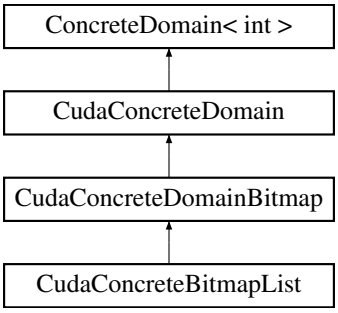
actual internal representation of domain.

The documentation for this class was generated from the following files:

- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/cuda_concrete_domain.h
- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/cuda_concrete_domain.cpp

6.10 CudaConcreteDomainBitmap Class Reference

Inheritance diagram for CudaConcreteDomainBitmap:



Public Member Functions

- CudaConcreteDomainBitmap (size_t size)
- CudaConcreteDomainBitmap (size_t size, int min, int max)
- unsigned int size () const

  It returns the current size of the domain.
- void shrink (int min, int max)
- void subtract (int value)
- void in_min (int min)
- void in_max (int max)
- void add (int value)
- void add (int min, int max)
- bool contains (int value) const
- bool is_singleton () const
- int get_singleton () const
- const void * get_representation () const
- void print () const

Static Protected Member Functions

- static constexpr int IDX_CHUNK (int val)
- static constexpr int IDX_BIT (int val)
- static constexpr int NUM_CHUNKS (int size)

Protected Attributes

- unsigned int _num_valid_bits

  Number of bits set to 1.

Static Protected Attributes

- static constexpr int BITS_IN_BYTE = INT8_C(8)
- static constexpr int BITS_IN_CHUNK = sizeof(int) * BITS_IN_BYTE

Additional Inherited Members

6.10.1 Constructor & Destructor Documentation

6.10.1.1 CudaConcreteDomainBitmap::CudaConcreteDomainBitmap (size_t *size*)

Constructor for CudaConcreteDomainBitmap.

Parameters

size	the size in bytes to allocate for the bitmap.

Note

the bitmap is represented considering lower bound = 0 and upper bound given by the parameter size. initially all bits are set to 1 (i.e. valid bits).

6.10.1.2 CudaConcreteDomainBitmap::CudaConcreteDomainBitmap (size_t *size*, int *min*, int *max*)

Constructor for CudaConcreteDomainBitmap.

Parameters

size	the size in bytes to allocate for the bitmap.
min	lower bound for {min, max} set initilization. min must be greater than or equal to 0 and less than or equal to the max number of bits storable using size bytes.
max	upper bound for {min, max} set initilization. max must be less than or equal to max number of bits storable using size bytes and greater than or equal to 0.

Note

the bitmap is represented considering lower bound = 0 and upper bound given by the parameter size. initially all bits in {min, max} are set to 1 (i.e. valid bits).

6.10.2 Member Function Documentation

6.10.2.1 void CudaConcreteDomainBitmap::add (int *value*) [virtual]

It computes union of this domain and {value}.

Parameters

value	it specifies the value which is being added.

Note

value is given w.r.t. a lower bound of 0.

Implements ConcreteDomain< int >.

Reimplemented in CudaConcreteBitmapList.

6.10.2.2 void CudaConcreteDomainBitmap::add (int *min,* int *max*) [virtual]

It computes union of this domain and {min, max}.

Parameters

min	lower bound of the new domain which is being added.
max	upper bound of the new domain which is being added.

Todo implement using checks on chunks of bits (i.e. sublinear cost).

Implements ConcreteDomain< int >.

Reimplemented in CudaConcreteBitmapList.

6.10.2.3 bool CudaConcreteDomainBitmap::contains (int *value*) const [virtual]

It checks whether the value belongs to the domain or not.

Parameters

value	to check whether it is in the current domain.

Note

value is given w.r.t. the lower bound of 0.

Implements ConcreteDomain< int >.

Reimplemented in CudaConcreteBitmapList.

6.10.2.4 const void * CudaConcreteDomainBitmap::get_representation ( ) const [virtual]

It returns a void pointer to an object representing the current representation of the domain (e.g., bitmap).

Returns

void pointer to the concrete domain representation.

Implements ConcreteDomain< int >.

6.10.2.5 int CudaConcreteDomainBitmap::get_singleton ( ) const [virtual]

It returns the value of the domain element if it is a singleton.

Returns

the value of the singleton element.

Note

it throws an exception if domain is not singleton.

Implements ConcreteDomain< int >.

6.10.2.6 static constexpr int CudaConcreteDomainBitmap::IDX_BIT (int *val*) [inline], [static], [protected]

Get index of the bit that represents the value val module the size of a chuck, i.e., the position of the corresponding bit within a chunk.

Parameters

val	the value w.r.t. the function calculates its position within a chunk of bits

Returns

position (starting from 0) of the bit corresponding to val.

6.10.2.7 static constexpr int CudaConcreteDomainBitmap::IDX_CHUNK (int *val*) [inline], [static], [protected]

Get index of the chunk of bits containing the bit representing the value given in input.

Parameters

max	lower bound used to calculated the index of the bitmap

Returns

number of int used as bitmaps to represent max

6.10.2.8 void CudaConcreteDomainBitmap::in_max (int *max*) [virtual]

It updates the domain according to max value.

Parameters

max	domain value.

Implements ConcreteDomain< int >.

Reimplemented in CudaConcreteBitmapList.

6.10.2.9 void CudaConcreteDomainBitmap::in_min (int *min*) [virtual]

It updates the domain according to min value.

Parameters

min	domain value.

Implements ConcreteDomain< int >.

Reimplemented in CudaConcreteBitmapList.

6.10.2.10 bool CudaConcreteDomainBitmap::is_singleton ( ) const [virtual]

It checks whether the current domain contains only an element (i.e., it is a singleton).

Returns

true if the current domain is singleton, false otherwise.

Implements ConcreteDomain< int >.

6.10.2.11 static constexpr int CudaConcreteDomainBitmap::NUM_CHUNKS (int *size*) [inline], [static], [protected]

Get the number of chunks needed to represent a domain of size values.

Parameters

size	the size in terms of number of elements of the domain to represent as bitmap.

Returns

number of chunks needed to represent size valus.

6.10.2.12 void CudaConcreteDomainBitmap::print ( ) const [virtual]

It prints the current domain representation (its state).

Note

it prints the content of the object given by "get_representation ()".

Implements ConcreteDomain< int >.

Reimplemented in CudaConcreteBitmapList.

6.10.2.13 void CudaConcreteDomainBitmap::shrink (int *min*, int *max*) [virtual]

It updates the domain to have values only within min/max.

Parameters

min	new lower bound to set for the current domain.
max	new upper bound to set for the current domain.

Implements ConcreteDomain< int >.

Reimplemented in CudaConcreteBitmapList.

6.10.2.14 void CudaConcreteDomainBitmap::subtract (int *value*) [virtual]

It substracts {value} from the current domain.

Parameters

value	the value to subtract from the current domain.

Implements ConcreteDomain< int >.

Reimplemented in CudaConcreteBitmapList.

6.10.3 Member Data Documentation

6.10.3.1 constexpr int CudaConcreteDomainBitmap::BITS_IN_BYTE = INT8_C(8) [static], [protected]

Macro for the size of a byte in terms of bits.

6.10.3.2 constexpr int CudaConcreteDomainBitmap::BITS_IN_CHUNK = sizeof(int) * BITS_IN_BYTE [static], [protected]

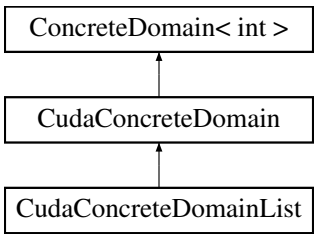
Macro for the size of a chunk in terms of bits.

The documentation for this class was generated from the following files:

- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/cuda_concrete_bitmap.h
- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/cuda_concrete_bitmap.cpp

6.11 CudaConcreteDomainList Class Reference

Inheritance diagram for CudaConcreteDomainList:



Public Member Functions

- CudaConcreteDomainList (size_t size, int min, int max)
- unsigned int size () const

  It returns the current size of the domain.
- void shrink (int min, int max)
- void subtract (int value)
- void in_min (int min)
- void in_max (int max)
- void add (int value)
- void add (int min, int max)
- bool contains (int val) const
- bool is_singleton () const
- int get_singleton () const
- const void * get_representation () const
- void print () const

Protected Member Functions

- int find_pair (int val) const
- int find_prev_pair (int val) const
- int find_next_pair (int val) const

Protected Attributes

- int _num_pairs

  Number of pairs in the list (list size)
- int _max_allowed_pairs

  Max number of storable pairs in the concrete domain.
- unsigned int _domain_size

6.11.1 Constructor & Destructor Documentation

6.11.1.1 CudaConcreteDomainList::CudaConcreteDomainList (size_t *size*, int *min*, int *max*)

Constructor for CudaConcreteDomainList.

Parameters

size	the size in bytes to allocate for the bitmap.
min	lower bound in {min, max}
max	upper bound in {min, max}

6.11.2 Member Function Documentation

6.11.2.1 void CudaConcreteDomainList::add (int *value*) [virtual]

It computes union of this domain and {value}.

Parameters

value	it specifies the value which is being added.

Implements ConcreteDomain< int >.

6.11.2.2 void CudaConcreteDomainList::add (int *min*, int *max*) [virtual]

It computes union of this domain and {min, max}.

Parameters

min	lower bound of the new domain which is being added.
max	upper bound of the new domain which is being added.

Implements ConcreteDomain< int >.

6.11.2.3 bool CudaConcreteDomainList::contains (int *val*) const [virtual]

It checks whether the value belongs to the domain or not.

Parameters

val	to check whether it is in the current domain.

Note

val is given w.r.t. the lower bound of 0.

Implements ConcreteDomain< int >.

6.11.2.4 int CudaConcreteDomainList::find_next_pair (int *val*) const [protected]

Find the index of the first pair with values greater than val.

Parameters

val	to be compared in the list of pairs.

Returns

the index of the pair with val greater than val, -1 if no such pair exists.

6.11.2.5 int CudaConcreteDomainList::find_pair (int *val*) const [protected]

Find the index of the pair containing val.

Parameters

val	to be searched in the list of pairs.

Returns

the index of the pair containing val, -1 otherwise.

6.11.2.6 int CudaConcreteDomainList::find_prev_pair (int *val*) const [protected]

Find the index of the last pair with values smaller than val.

Parameters

val	to be compared in the list of pairs.

Returns

the index of the pair with val lower than val, -1 if no such pair exists.

6.11.2.7 const void * CudaConcreteDomainList::get_representation ( ) const [virtual]

It returns a void pointer to an object representing the current representation of the domain (e.g., bitmap).

Returns

void pointer to the concrete domain representation.

Implements ConcreteDomain< int >.

6.11.2.8 int CudaConcreteDomainList::get_singleton ( ) const [virtual]

It returns the value of type T of the domain if it is a singleton.

Returns

the value of the singleton element.

Note

it throws an exception if domain is not singleton.

Implements ConcreteDomain< int >.

6.11.2.9 void CudaConcreteDomainList::in_max (int *max*) [virtual]

It updates the domain according to max value.

Parameters

max	domain value.

Implements ConcreteDomain< int >.

6.11.2.10 void CudaConcreteDomainList::in_min (int *min*) [virtual]

It updates the domain according to min value.

Parameters

min	domain value.

Implements ConcreteDomain< int >.

6.11.2.11 bool CudaConcreteDomainList::is_singleton ( ) const [virtual]

It checks whether the current domain contains only an element (i.e., it is a singleton).

Returns

true if the current domain is singleton, false otherwise.

Implements ConcreteDomain< int >.

6.11.2.12 void CudaConcreteDomainList::print ( ) const [virtual]

It prints the current domain representation (its state).

Note

it prints the content of the object given by "get_representation ()" .

Implements ConcreteDomain< int >.

6.11.2.13 void CudaConcreteDomainList::shrink (int *min*, int *max*) [virtual]

It updates the domain to have values only within min/max.

Parameters

min	new lower bound to set for the current domain.
max	new upper bound to set for the current domain.

Implements ConcreteDomain< int >.

6.11.2.14 void CudaConcreteDomainList::subtract (int *value*) [virtual]

It substracts {value} from the current domain.

Parameters

value	the value to subtract from the current domain.

Note

a value is removed only if it corresponds to a lower/upper bound.

Implements ConcreteDomain< int >.

6.11.3 Member Data Documentation

6.11.3.1 unsigned int CudaConcreteDomainList::_domain_size [protected]

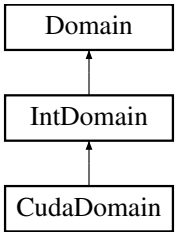
Current domain size, i.e., sum of the elements on each pair of bounds in the list.

The documentation for this class was generated from the following files:

- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/cuda_concrete_list.h
- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/cuda_concrete_list.cpp

6.12 CudaDomain Class Reference

Inheritance diagram for CudaDomain:



Public Member Functions

- DomainPtr clone () const

  Clone the current domain and returns a pointer to it.
- void init_domain (int min, int max)
- size_t get_allocated_bytes () const
- EventType get_event () const

  Get event on the current domain.
- size_t get_size () const
- int lower_bound () const

  Get the domain's lower bound.
- int upper_bound () const

  Get the domain's upper bound.
- void set_bounds (int min, int max)
- void shrink (int min, int max)
- bool set_singleton (int)

  Set domain as singleton.
- bool subtract (int n)

  Subtract the element from the domain (see int_domain.h)
- void add_element (int n)
- void in_min (int min)

  Increase the lower_bound to min (see int_domain.h)
- void in_max (int max)

  Decrease the upper_bound to max (see int_domain.h)
- void print () const

  Print info about domain.
- void print_domain () const

  Print internal domain representation.

Protected Member Functions

- DomainPtr clone_impl () const

  Clone method to clone the current object.
- EventType int_to_event () const

  Convert the current event int to a domain event.

- void event_to_int (EventType evt) const

  Convert a domain event to the current integer.
- void set_bit_representation ()

  Switch to bitmap representation of domain.
- void set_bitlist_representation (int num_list=INT_BITLIST)

  Switch to list representation of domain.
- void set_list_representation (int num_list=INT_LIST)

  Switch to list representation of domain.
- CudaDomainRepresenation get_representation () const

  Get domain representation (i.e., bitmap, bitmaplist, or list)
- void switch_list_to_bitmaplist ()

Static Protected Member Functions

- static constexpr int EVT_IDX ()
- static constexpr int **REP_IDX** ()
- static constexpr int **LB_IDX** ()
- static constexpr int **UB_IDX** ()
- static constexpr int **DSZ_IDX** ()
- static constexpr int **BIT_IDX** ()
- static constexpr int IDX_CHUNK (int val)
- static constexpr int IDX_BIT (int val)
- static int num_chunks (int n)

Protected Attributes

- CudaConcreteDomainPtr _concrete_domain
- int * _domain
- size_t _num_allocated_bytes
- size_t _num_int_chunks

Static Protected Attributes

- static constexpr int **INT_BITMAP** = 0
- static constexpr int **INT_BITLIST** = -1
- static constexpr int **INT_LIST** = 1
- static constexpr int BITS_IN_BYTE = INT8_C(8)
- static constexpr int SHARED_MEM_KB = 47
- static constexpr size_t MAX_BYTES_SIZE = SHARED_MEM_KB * 1024
- static constexpr size_t MAX_STATUS_SIZE = 5 * sizeof(int)
- static constexpr size_t MAX_DOMAIN_VALUES = ((MAX_BYTES_SIZE - MAX_STATUS_SIZE) / sizeof(int))

Additional Inherited Members

6.12.1 Member Function Documentation

6.12.1.1 void CudaDomain::add_element (int *n*) [virtual]

Add an element val to the current domain (see int_domain.h).

Note

if the element is out of the current bounds, no element will be added, i.e., the domain mantains the current size.

Implements IntDomain.

6.12.1.2 static constexpr int CudaDomain::EVT_IDX ( ) [inline], [static], [protected]

Constants used to retrieve the current domain description. Domain represented as: | EVT | REP | LB | UB | DSZ || ... BIT ... |. See system_description.h.

6.12.1.3 size_t CudaDomain::get_allocated_bytes ( ) const

Get the number of allocated bytes needed for representing the current domain w.r.t. its lower and upper bounds.

Returns

the number of allocated bytes.

6.12.1.4 size_t CudaDomain::get_size ( ) const [virtual]

Get domain size. It returns the currenst size of the domain, checking whether there are "holes" according to the current representation of the domain (i.e., bitmap or list):

Returns

the current domain's size.

Implements Domain.

6.12.1.5 static constexpr int CudaDomain::IDX_BIT (int *val*) [inline], [static], [protected]

Get index of the last int used as bitmap to represent [min, max].

Parameters

max	lower bound used to calculated the index of the bitmap

Returns

number of int used as bitmaps to represent max

6.12.1.6 static constexpr int CudaDomain::IDX_CHUNK (int *val*) [inline], [static], [protected]

Get index of the chunk of bits containing the bit representing the value given in input.

Parameters

max	lower bound used to calculated the index of the bitmap

Returns

number of int used as bitmaps to represent max

6.12.1.7 void CudaDomain::init_domain (int *min*, int *max*) [virtual]

Initializes domain with default values:

- Event: no event;
- Representation: list or bitmap according to [min, max];
- Lower bound: min;
- Upper bound: max;
- Size: |max - min + 1| or MAX_INT if max = MAN_INT()/2 and min = MIN_INT() / 2, etc..

Note

It instantiate an array of ints of at most MAX_BYTES_SIZE.

Parameters

min	lower bound of the domain
max	upper bound of the domain

Returns

it fails whenever consistency check on min/max fails (i.e., max $<$ min).

Implements IntDomain.

6.12.1.8 static int CudaDomain::num_chunks (int *n*) [inline], [static], [protected]

Return the number of 32-bit integers needed to represent a set of n domain's values.

Parameters

n	number of values to represent as bits

Returns

number of 32-bit integer chunks needed to represent n values.

6.12.1.9 void CudaDomain::set_bounds (int *min*, int *max*)

The same as set_bounds. It shrinks the domain to {min, max}.

Parameters

min	lower bound
max	upper bound

6.12.1.10 void CudaDomain::shrink (int *min*, int *max*) [virtual]

It specializes the parent method in order to set up the array of (int) values. It istantiates a domain [min, max]. This actually updates the bounds and it performs consistency checking and updating of the domain size.

Parameters

min	lower bound
max	upper bound

Implements IntDomain.

6.12.1.11 void CudaDomain::switch_list_to_bitmaplist ( ) [protected]

Take the current list representation and switch it to a bitmap list represenatation.

6.12.2 Member Data Documentation

6.12.2.1 CudaConcreteDomainPtr CudaDomain::_concrete_domain [protected]

Actual domain is represented by an object of type "cuda_concrete_domain". This domain can be a either bitmap, a list of bounds, or a bitmap list, depending on the size of the domain. Internal switches between domain representations are performed automatically as soon as the domain's size is reduced to a given threshold.

Note

system_description.h

6.12.2.2 int* CudaDomain::_domain [protected]

Domain is the actual bit domain representation. Operations are performed on _concrete_domain, status is stored on _domain. When another class needs this domain's representation, _domain will be returned.

6.12.2.3 size_t CudaDomain::_num_allocated_bytes [protected]

Total allocated bytes for representing the current domain.

6.12.2.4 size_t CudaDomain::_num_int_chunks [protected]

Total number of bitchunks.

Note

it does not consider the first part related to information about domain.

6.12.2.5 constexpr int CudaDomain::BITS_IN_BYTE = INT8_C(8) [static], [protected]

Macro to use for declaring the size of a byte in terms of bits.

6.12.2.6 constexpr size_t CudaDomain::MAX_BYTES_SIZE = SHARED_MEM_KB * 1024 [static], [protected]

Maximum domain size in terms of bytes.

Note

see CUDA specifications. Usually, (48 - 1) kB = 47 * 1024 = 48128 Byte.

6.12.2.7 constexpr size_t CudaDomain::MAX_DOMAIN_VALUES = ((MAX_BYTES_SIZE - MAX_STATUS_SIZE) / sizeof(int)) [static], [protected]

Maximum size in terms of storable values. Worst case: list of type {1, 1}, {3, 3}, {5, 5}, ... Number of integers = ((MAX_BYTES_SIZE - 5 ∗ sizeof(int)) / sizeof(int))

Note

see CUDA specifications.

6.12.2.8 constexpr size_t CudaDomain::MAX_STATUS_SIZE = 5 ∗ sizeof(int) [static], [protected]

Number of Bytes needed for representing the current domain status.

6.12.2.9 constexpr int CudaDomain::SHARED_MEM_KB = 47 [static], [protected]

Shared memory available.

Note

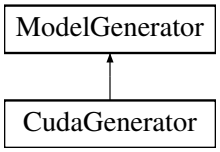
keep 1 kB less than the actual memory available.

The documentation for this class was generated from the following files:

- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/cuda_domain.h
- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/cuda_domain.cpp

6.13 CudaGenerator Class Reference

Inheritance diagram for CudaGenerator:



Public Member Functions

- VariablePtr get_variable (TokenPtr)
  See "model_generator.h".
- ConstraintPtr get_constraint (TokenPtr)
  See "model_generator.h".
- SearchEnginePtr get_search_engine (TokenPtr)
  See "model_generator.h".

Protected Attributes

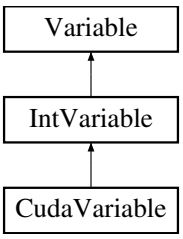
- std::string **_dbg**

The documentation for this class was generated from the following files:

- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/cuda_model_generator.h
- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/cuda_model_generator.cpp

6.14 CudaVariable Class Reference

Inheritance diagram for CudaVariable:



Public Member Functions

- CudaVariable ()
- CudaVariable (int idv)
- void set_domain ()
- void set_domain (int lw, int ub)
- void set_domain (std::vector< std::vector< int > > elems)
- void print () const

  print info about the current domain

Additional Inherited Members

6.14.1 Constructor & Destructor Documentation

6.14.1.1 CudaVariable::CudaVariable ( )

Base constructor: create a variable with new id. The id is given by a global id generator.

6.14.1.2 CudaVariable::CudaVariable (int *idv*)

One parameter constructor: create a variable with a given id.

Parameters

idv	identifier to give to the variable

6.14.2 Member Function Documentation

6.14.2.1 void CudaVariable::set_domain ( ) [virtual]

Set domain's bounds. If no bounds are provided, an unbounded domain (int) is istantiated. If an array of elements A is provided, the function instantiates a domain D = [min A, max A], deleting all the elements d in D s.t. d does not belong to A.

Implements IntVariable.

6.14.2.2 void CudaVariable::set_domain (int *lw*, int *ub*) [virtual]

Set domain's bounds. A new domain [lw, ub] is generated.

Parameters

lw	lower bound
ub	upper bound

Implements IntVariable.

6.14.2.3 void CudaVariable::set_domain (std::vector< std::vector< int > > *elems*) [virtual]

Set domain's elements. A domain {d_1, ..., d_n} is generated.

Parameters

elems	vector of vectors (subsets) of domain's elements

Todo implement set of sets of elements.

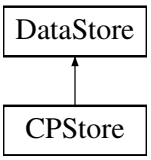
Implements IntVariable.

The documentation for this class was generated from the following files:

- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/cuda_variable.h
- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/cuda_variable.cpp

6.15 DataStore Class Reference

Inheritance diagram for DataStore:



Public Member Functions

- virtual bool load_model (std::string="")=0
- virtual void init_model ()=0

  Init model using the information read from files.
- virtual void print_model_info ()=0

  Print info about the model.
- virtual CPModel * get_model ()

  Get the instantiated model.
- virtual void **print_model_variable_info** ()
- virtual void **print_model_domain_info** ()
- virtual void **print_model_constraint_info** ()

Protected Member Functions

- DataStore (std::string in_file)

Protected Attributes

- bool **_timer**
- bool **_verbose**
- std::string **_dbg**
- std::string **_in_file** = ""
- CPModel * _cp_model

  CP Model.

6.15.1 Constructor & Destructor Documentation

6.15.1.1 DataStore::DataStore (std::string *in_file*) [protected]

Constructor.

Parameters

in_file	file path of the model to parse.

6.15.2 Member Function Documentation

6.15.2.1 virtual bool DataStore::load_model (std::string = " ") [pure virtual]

Load model from input file (FlatZinc model).

Note

: the model described as a set of tokens is stored in the Tokenization class used by the parser. The parser has access to the set of tokens and it manages them in order to retrieve the correct set of tokens to initialize variables, and constraints. See Parser interface.

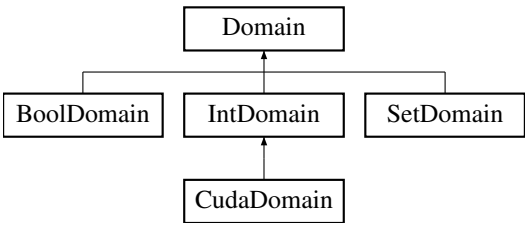
Implemented in CPStore.

The documentation for this class was generated from the following files:

- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/data_store.h
- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/data_store.cpp

6.16 Domain Class Reference

Inheritance diagram for Domain:



Public Member Functions

- void set_type (DomainType dt)

- DomainType **get_type** () const
- virtual DomainPtr clone () const =0

  Clone the current domain and returns a pointer to it.
- virtual EventType get_event () const =0

  Get the current event on the domain.
- virtual size_t get_size () const =0

  Returns the size of the domain.
- virtual bool is_empty () const =0

  Returns true if the domain is empty.
- virtual bool is_singleton () const =0

  Returns true if the domain has only one element.
- virtual void print () const =0

  Print info about the current domain.

Static Public Member Functions

- static constexpr int MIN_DOMAIN ()

  Constants for int min/max domain bounds.
- static constexpr int MAX_DOMAIN ()

  Constants for int min/max domain bounds.

Protected Attributes

- std::string **_dbg**
- DomainType **_dom_type**

6.16.1 Member Function Documentation

6.16.1.1 void Domain::set_type (DomainType *dt*)

Set domain's type (use get_type to get the type).

Parameters

dt	domain type of type DomainType

The documentation for this class was generated from the following files:

- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/domain.h
- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/domain.cpp

6.17 Event Class Reference

Public Member Functions

- **Event** (EventType domain_event)
- virtual EventType **get_domain_event** () const

Protected Attributes

- EventType **_domain_event**

The documentation for this class was generated from the following files:

- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/event.h
- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/event.cpp

6.18 FactoryModelGenerator Class Reference

Static Public Member Functions

- static ModelGenerator * get_generator (GeneratorType gt)

  Get the right instance of a generator based on the input.

The documentation for this class was generated from the following file:

- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/factory_generator.h

6.19 FactoryParser Class Reference

Static Public Member Functions

- static Parser * get_parser (ParserType pt)

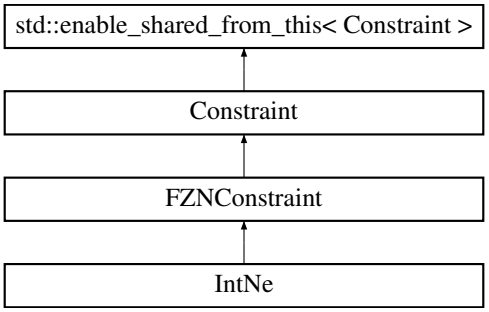
  Get the right parser based on the input.

The documentation for this class was generated from the following file:

- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/factory_parser.h

6.20 FZNConstraint Class Reference

Inheritance diagram for FZNConstraint:



Public Member Functions

- virtual void setup (std::vector< VariablePtr > vars, std::vector< std::string > args)=0
- void attach_me () override

- void consistency () override
- bool satisfied () override
- void remove_constraint ()
- void print () const override

  Prints info.
- void print_semantic () const override

  Prints the semantic of this constraint.

Static Public Member Functions

- static FZNConstraintType int_to_type (int number_id)
- static int type_to_int (FZNConstraintType c_type)
- static int name_to_id (std::string c_name)

Static Public Attributes

- static const std::string **ARRAY_BOOL_AND** = "array_bool_and"
- static const std::string **ARRAY_BOOL_ELEMENT** = "array_bool_element"
- static const std::string **ARRAY_BOOL_OR** = "array_bool_or"
- static const std::string **ARRAY_FLOAT_ELEMENT** = "array_float_element"
- static const std::string **ARRAY_INT_ELEMENT** = "array_int_element"
- static const std::string **ARRAY_SET_ELEMENT** = "array_set_element"
- static const std::string **ARRAY_VAR_BOOL_ELEMENT** = "array_var_bool_element"
- static const std::string **ARRAY_VAR_FLOAT_ELEMENT** = "array_var_float_element"
- static const std::string **ARRAY_VAR_INT_ELEMENT** = "array_var_int_element"
- static const std::string **ARRAY_VAR_SET_ELEMENT** = "array_var_set_element"
- static const std::string **BOOL2INT** = "bool2int"
- static const std::string **BOOL_AND** = "bool_and"
- static const std::string **BOOL_CLAUSE** = "bool_clause"
- static const std::string **BOOL_EQ** = "bool_eq"
- static const std::string **BOOL_EQ_REIF** = "bool_eq_reif"
- static const std::string **BOOL_LE** = "bool_le"
- static const std::string **BOOL_LE_REIF** = "bool_le_reif"
- static const std::string **BOOL_LT** = "bool_lt"
- static const std::string **BOOL_LT_REIF** = "bool_lt_reif"
- static const std::string **BOOL_NOT** = "bool_not"
- static const std::string **BOOL_OR** = "bool_or"
- static const std::string **BOOL_XOR** = "bool_xor"
- static const std::string **FLOAT_ABS** = "float_abs"
- static const std::string **FLOAT_ACOS** = "float_acos"
- static const std::string **FLOAT_ASIN** = "float_asin"
- static const std::string **FLOAT_ATAN** = "float_atan"
- static const std::string **FLOAT_COS** = "float_cos"
- static const std::string **FLOAT_COSH** = "float_cosh"
- static const std::string **FLOAT_EXP** = "float_exp"
- static const std::string **FLOAT_LN** = "float_ln"
- static const std::string **FLOAT_LOG10** = "float_log10"
- static const std::string **FLOAT_LOG2** = "float_log2"
- static const std::string **FLOAT_SQRT** = "float_sqrt"
- static const std::string **FLOAT_SIN** = "float_sin"
- static const std::string **FLOAT_SINH** = "float_sinh"
- static const std::string **FLOAT_TAN** = "float_tan"
- static const std::string **FLOAT_TANH** = "float_tanh"

- static const std::string **FLOAT_EQ** = "float_eq"
- static const std::string **FLOAT_EQ_REIF** = "float_eq_reif"
- static const std::string **FLOAT_LE** = "float_le"
- static const std::string **FLOAT_LE_REIF** = "float_le_reif"
- static const std::string **FLOAT_LIN_EQ** = "float_lin_eq"
- static const std::string **FLOAT_LIN_EQ_REIF** = "float_lin_eq_reif"
- static const std::string **FLOAT_LIN_LE** = "float_lin_le"
- static const std::string **FLOAT_LIN_LE_REIF** = "float_lin_le_reif"
- static const std::string **FLOAT_LIN_LT** = "float_lin_lt"
- static const std::string **FLOAT_LIN_LT_REIF** = "float_lin_lt_reif"
- static const std::string **FLOAT_LIN_NE** = "float_lin_ne"
- static const std::string **FLOAT_LIN_NE_REIF** = "float_lin_ne_reif"
- static const std::string **FLOAT_LT** = "float_lt"
- static const std::string **FLOAT_LT_REIF** = "float_lt_reif"
- static const std::string **FLOAT_MAX** = "float_max"
- static const std::string **FLOAT_MIN** = "float_min"
- static const std::string **FLOAT_NE** = "float_ne"
- static const std::string **FLOAT_NE_REIF** = "float_ne_reif"
- static const std::string **FLOAT_PLUS** = "float_plus"
- static const std::string **INT_ABS** = "int_abs"
- static const std::string **INT_DIV** = "int_div"
- static const std::string **INT_EQ** = "int_eq"
- static const std::string **INT_EQ_REIF** = "int_eq_reif"
- static const std::string **INT_LE** = "int_le"
- static const std::string **INT_LE_REIF** = "int_le_reif"
- static const std::string **INT_LIN_EQ** = "int_lin_eq"
- static const std::string **INT_LIN_EQ_REIF** = "int_lin_eq_reif"
- static const std::string **INT_LIN_LE** = "int_lin_le"
- static const std::string **INT_LIN_LE_REIF** = "int_lin_le_reif"
- static const std::string **INT_LIN_NE** = "int_lin_ne"
- static const std::string **INT_LIN_NE_REIF** = "int_lin_ne_reif"
- static const std::string **INT_MAX_C** = "int_max"
- static const std::string **INT_MIN_C** = "int_min"
- static const std::string **INT_MOD** = "int_mod"
- static const std::string **INT_NE** = "int_ne"
- static const std::string **INT_NE_REIF** = "int_ne_reif"
- static const std::string **INT_PLUS** = "int_plus"
- static const std::string **INT_TIMES** = "int_times"
- static const std::string **INT2FLOAT** = "int2float"
- static const std::string **SET_CARD** = "set_card"
- static const std::string **SET_DIFF** = "set_diff"
- static const std::string **SET_EQ** = "set_eq"
- static const std::string **SET_EQ_REIF** = "set_eq_reif"
- static const std::string **SET_IN** = "set_in"
- static const std::string **SET_IN_REIF** = "set_in_reif"
- static const std::string **SET_INTERSECT** = "set_intersect"
- static const std::string **SET_LE** = "set_le"
- static const std::string **SET_LT** = "set_lt"
- static const std::string **SET_NE** = "set_ne"
- static const std::string **SET_NE_REIF** = "set_ne_reif"
- static const std::string **SET_SUBSET** = "set_subset"
- static const std::string **SET_SUBSET_REIF** = "set_subset_reif"
- static const std::string **SET_SYMDIFF** = "set_symdiff"
- static const std::string **SET_UNION** = "set_union"
- static const std::string **OTHER** = "other"

Protected Member Functions

- virtual void set_events (EventType event=EventType::CHANGE_EVT)
- FZNConstraint (std::string name)

Protected Attributes

- FZNConstraintType _constraint_type

  FlatZinc constraint type.
- int _scope_size

  Scope size.

6.20.1 Constructor & Destructor Documentation

6.20.1.1 FZNConstraint::FZNConstraint (std::string *name*) [protected]

Base constructor.

Parameters

name	the name of the FlatZinc constraint.
vars	the vector of (shared) pointers to the variables in the scope of this constraint.
args	the vector of auxiliary arguments stored as strings needed by this constraint in order to be propagated.

Note

FZNConstraint instantiated with this constructor need to be defined in terms of variables in their scope and, if needed, auxiliary parameters.

6.20.2 Member Function Documentation

6.20.2.1 void FZNConstraint::attach_me ( ) [override], [virtual]

It attaches this constraint (observer) to the list of the variables in its scope. When a variable changes state, this constraint could be automatically notified (depending on the variable).

Implements Constraint.

6.20.2.2 void FZNConstraint::consistency ( ) [override], [virtual]

It is a (most probably incomplete) consistency function which removes the values from variable domains. Only values which do not have any support in a solution space are removed.

Implements Constraint.

Reimplemented in IntNe.

6.20.2.3 FZNConstraintType FZNConstraint::int_to_type (int *number_id*) [static]

It converts a number_id name to the correspondent FZNConstraintType type.

Parameters

number_id	the number id of the FlatZinc constraint.

Returns

the type of the FlatZinc constraint.

6.20.2.4 int FZNConstraint::name_to_id (std::string *c_name*) [static]

It converts a string representing the name of a constraint to a unique idetifier for the correspondent type of FlatZinc constraint.

Parameters

c_name	name of a FlatZinc constraint.

Returns

the number_id correspondent to name.

6.20.2.5 void FZNConstraint::remove_constraint ( ) [virtual]

It removes the constraint by removing this constraint from all variables in its scope.

Implements Constraint.

6.20.2.6 bool FZNConstraint::satisfied ( ) [override], [virtual]

It checks if the constraint is satisfied.

Returns

true if the constraint if for certain satisfied, false otherwise.

Note

If this function is incorrectly implementd, a constraint may not be satisfied in a solution.

Implements Constraint.

Reimplemented in IntNe.

6.20.2.7 void FZNConstraint::set_events (EventType *event* = EventType::CHANGE_EVT) [protected], [virtual]

Set the events that trigger this constraint.

Note

default: CHANGE_EVT.
different constraints should specilize this method with the appropriate list of events.

6.20.2.8 virtual void FZNConstraint::setup (std::vector< VariablePtr > *vars*, std::vector< std::string > *args*) [pure virtual]

It sets the variables and the arguments for this constraint.

Parameters

vars	a vector of pointers to the variables in the constraint's scope.
args	a vector of strings representing the auxiliary arguments needed by the constraint in order to ensure consistency.

Implemented in IntNe.

6.20.2.9 int FZNConstraint::type_to_int (FZNConstraintType *c_type*) [static]

It converts a FZNConstraintType to the correspondent integer type.

Parameters

c_type	the type of the FlatZinc constraint.

Returns

the number_id correspondent to c_type.

The documentation for this class was generated from the following files:

- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/fzn_constraint.h
- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/fzn_constraint.cpp

6.21 FZNConstraintFactory Class Reference

Static Public Member Functions

- static FZNConstraint * get_fzn_constraint (std::string c_name, std::vector< VariablePtr > vars, std::vector< std::string > args)
- static ConstraintPtr get_fzn_constraint_shr_ptr (std::string c_name, std::vector< VariablePtr > vars, std::vector< std::string > args)

6.21.1 Member Function Documentation

6.21.1.1 static FZNConstraint* FZNConstraintFactory::get_fzn_constraint (std::string *c_name,* std::vector< VariablePtr > *vars,* std::vector< std::string > *args*) [inline], [static]

Get the right instance of FlatZinc constraint according to its type described by the input string.

Parameters

c_name	the FlatZinc name of the constraint to instantiate.
vars	the vector of (shared) pointer to the FD variables in the scope of the constraint to instantiate.
args	the vector of strings representing the auxiliary arguments needed by the constraint to instantiate in order to be propagated.

6.21.1.2 static ConstraintPtr FZNConstraintFactory::get_fzn_constraint_shr_ptr (std::string *c_name,* std::vector< VariablePtr > *vars,* std::vector< std::string > *args*) [inline], [static]

Get the right instance of FlatZinc constraint according to its type described by the input string.

Parameters

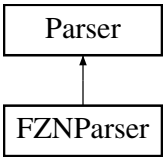
c_name	the FlatZinc name of the constraint to instantiate.
vars	the vector of (shared) pointer to the FD variables in the scope of the constraint to instantiate.
args	the vector of strings representing the auxiliary arguments needed by the constraint to instantiate in order to be propagated.

The documentation for this class was generated from the following file:

- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/fzn_constraint_generator.h

6.22 FZNParser Class Reference

Inheritance diagram for FZNParser:



Public Member Functions

- **FZNParser** (std::string ifile)
- bool more_variables () const

  Ask whether there are more variables to get.
- bool more_constraints () const

  Ask whether there are more constraits to get.
- bool more_search_engines () const

  Ask whether there are more search engines to get.
- TokenPtr get_variable ()
- TokenPtr get_constraint ()
- TokenPtr get_search_engine ()
- TokenPtr get_next_content ()

  Get next (pointer to) token (i.e., FlatZinc element)
- void print () const

  Print info about the parser.

Additional Inherited Members

6.22.1 Member Function Documentation

6.22.1.1 TokenPtr FZNParser::get_constraint ( ) [virtual]

Get a "constraint" token.

Returns

token pointer to a "constraint" token.

Implements Parser.

6.22.1.2 TokenPtr FZNParser::get_next_content ( ) [virtual]

Get next (pointer to) token (i.e., FlatZinc element)

Set position on file to the most recent position

Implements Parser.

6.22.1.3 TokenPtr FZNParser::get_search_engine ( ) [virtual]

Get a "search_engine" token.

Returns

token pointer to a "search_engine" token.

Implements Parser.

6.22.1.4 TokenPtr FZNParser::get_variable ( ) [virtual]

Get a "variable" token.

Returns

token pointer to a "variable" token.

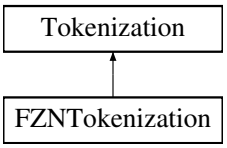
Implements Parser.

The documentation for this class was generated from the following files:

- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/fzn_parser.h
- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/fzn_parser.cpp

6.23 FZNTokenization Class Reference

Inheritance diagram for FZNTokenization:



Public Member Functions

- TokenPtr get_token ()

Additional Inherited Members

6.23.1 Member Function Documentation

6.23.1.1 TokenPtr FZNTokenization::get_token ( ) [virtual]

Specialized method: It actually gets the right token according to the FlatZinc format. Analysis is perfomed on "_c_token".

Implements Tokenization.

The documentation for this class was generated from the following files:

- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/fzn_tokenization.h
- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/fzn_tokenization.cpp

6.24 IdGenerator Class Reference

Public Member Functions

- void reset_int_id ()

  Reset id generator.
- void reset_str_id ()

  Reset id generator.
- void set_base_offset (int)

  Set (base) ids (if not already set)
- void set_base_prefix (std::string)

  Set (base) ids (if not already set)
- int **get_int_id** ()
- std::string **get_str_id** ()
- int **new_int_id** ()
- std::string **new_str_id** ()
- int **curr_int_id** ()
- std::string **curr_str_id** ()
- void **print_int_id** ()
- void **print_str_id** ()

Static Public Member Functions

- static IdGenerator * get_instance ()

  Constructor get (static) instance.

Protected Member Functions

- IdGenerator ()
- std::string n_to_str (int)

  Convert numbers to string.

6.24.1 Constructor & Destructor Documentation

6.24.1.1 IdGenerator::IdGenerator ( ) [protected]

Protected constructor: a client cannot instantiate Singleton directly.

The documentation for this class was generated from the following files:

- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/id_generator.h
- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/id_generator.cpp

6.25 InputData Class Reference

Public Member Functions

- bool **verbose** () const
- bool **timer** () const
- int **max_n_sol** () const
- std::string get_in_file () const

  Get input file (path to)
- std::string get_out_file () const

  Get output file (path to)

Static Public Member Functions

- static InputData * get_instance (int argc, char *argv[])

  Constructor get (static) instance.

Protected Member Functions

- InputData (int argc, char *argv[])

6.25.1 Constructor & Destructor Documentation

6.25.1.1 InputData::InputData (int *argc*, char * *argv[]*) [protected]

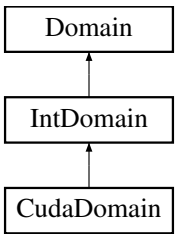
Protected constructor: a client cannot instantiate Singleton directly. Exit if the user did not set an input file!

The documentation for this class was generated from the following files:

- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/input_data.h
- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/input_data.cc

6.26 IntDomain Class Reference

Inheritance diagram for IntDomain:



Public Member Functions

- bool is_singleton () const

  Returns true if the domain has only one element.
- bool is_empty () const

  Returns true if the domain is empty.

- virtual void print () const

  Print base info about int domain.
- virtual int lower_bound () const =0

  Get the domain's lower bound.
- virtual int upper_bound () const =0

  Get the domain's upper bound.
- virtual void init_domain (int min, int max)=0
- virtual void shrink (int min, int max)=0
- virtual bool set_singleton (int val)=0
- virtual bool subtract (int val)=0
- virtual void add_element (int val)=0
- virtual void in_min (int min)=0
- virtual void in_max (int max)=0

Additional Inherited Members

6.26.1 Member Function Documentation

6.26.1.1 virtual void IntDomain::add_element (int *val*) [pure virtual]

It computes the union of the current domain with the domain represented by the singleton element given in input to the method. If the element is out of [lower_bound, upper_bound] it enlarges the domain.

Parameters

val	element to add to the current domain.

Implemented in CudaDomain.

6.26.1.2 virtual void IntDomain::in_max (int *max*) [pure virtual]

It updates the domain according to the maximum value.

Parameters

max	domain value.

Implemented in CudaDomain.

6.26.1.3 virtual void IntDomain::in_min (int *min*) [pure virtual]

It updates the domain according to the minimum value.

Parameters

min	domain value.

Implemented in CudaDomain.

6.26.1.4 virtual void IntDomain::init_domain (int *min*, int *max*) [pure virtual]

Initialize domain: this function is used to set up the domain as soon it is created. Classes that derive IntDomain specilize this method according to their internal representation of domain.

Implemented in CudaDomain.

6.26.1.5 virtual bool IntDomain::set_singleton (int *val*) [pure virtual]

Set domain to the singleton element given in input.

Parameters

val	the value to set as singleton

Returns

true if the domain has been set to singleton, false otherwise.

Implemented in CudaDomain.

6.26.1.6 virtual void IntDomain::shrink (int *min*, int *max*) [pure virtual]

Set domain's bounds. It updates the domain to have values only within the interval min..max.

Note

it does not update _lower_bound and _upper_bound here for efficiency reasons.

Parameters

lower	lower bound value
upper	upper bound value

Implemented in CudaDomain.

6.26.1.7 virtual bool IntDomain::subtract (int *val*) [pure virtual]

It intersects with the domain which is a complement of the value given as input, i.e., subtract a value from the current domain.

Parameters

val	the value to subtract from the current domain

Returns

true if succeed, false otherwise.

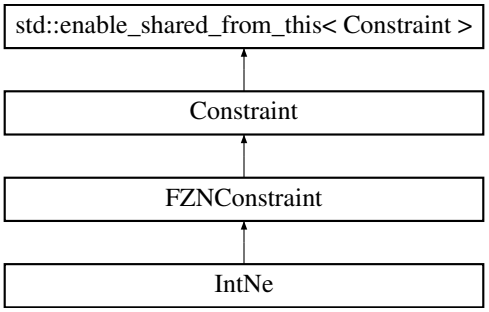
Implemented in CudaDomain.

The documentation for this class was generated from the following files:

- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/int_domain.h
- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/int_domain.cpp

6.27 IntNe Class Reference

Inheritance diagram for IntNe:



Public Member Functions

- IntNe ()
- IntNe (std::vector< VariablePtr > vars, std::vector< std::string > args)
- IntNe (int x, int y)
- IntNe (IntVariablePtr x, int y)
- IntNe (int x, IntVariablePtr y)
- IntNe (IntVariablePtr x, IntVariablePtr y)
- void setup (std::vector< VariablePtr > vars, std::vector< std::string > args)

  Setup method, see fzn_constraint.h.
- const std::vector< VariablePtr > scope () const
- void consistency () override

  It performs domain consistency.
- bool satisfied () override

  It checks if x != y.
- void print_semantic () const override

  Prints the semantic of this constraint.

Additional Inherited Members

6.27.1 Constructor & Destructor Documentation

6.27.1.1 IntNe::IntNe ( )

Basic constructor.

Note

after this constructor the client should call the setup method to setup the variables and parameters needed by the constraint.

6.27.1.2 IntNe::IntNe (std::vector< VariablePtr > *vars*, std::vector< std::string > *args*)

Basic constructor.

Note

this constructor implicitly calls the setup method to setup variables and arguments for this constraint.

6.27.1.3 IntNe::IntNe (int *x*, int *y*)

Basic constructor: it checks if x != y.

Parameters

x	an integer value.
y	an integer value.

6.27.1.4 IntNe::IntNe (IntVariablePtr *x*, int *y*)

Constructor.

Parameters

x	(pointer to) a FD variable.
y	an integer value.

Note

It subtracts the value y from the domain of the variable x if x has a domain defined on integers.

6.27.1.5 IntNe::IntNe (int *x*, IntVariablePtr *y*)

Constructor.

Parameters

x	an integer value.
y	(pointer to) a FD variable.

Note

It subtracts the value x from the domain of the variable y if y has a domain defined on integers.

6.27.1.6 IntNe::IntNe (IntVariablePtr *x*, IntVariablePtr *y*)

Constructor.

Parameters

x	(pointer to) a FD variable.
y	(pointer to) a FD variable.

6.27.2 Member Function Documentation

6.27.2.1 const std::vector< VariablePtr > IntNe::scope ( ) const [virtual]

It returns the vector of (shared) pointers of all the variables involved in a given constraint (i.e., its scope).

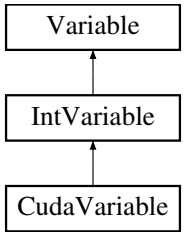
Implements Constraint.

The documentation for this class was generated from the following files:

- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/int_ne.h
- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/int_ne.cpp

6.28 IntVariable Class Reference

Inheritance diagram for IntVariable:



Public Member Functions

- virtual void set_domain ()=0
- virtual void set_domain (int lw, int ub)=0
- virtual void set_domain (std::vector< std::vector< int > > elems)=0
- virtual const DomainPtr domain ()

  Get (const) reference to this domain.
- virtual EventType get_event () const

  Get event on this domain.
- void set_domain_type (DomainType dt)
- size_t get_size () const
- bool is_singleton () const
- bool is_empty () const
- virtual int min () const
- virtual int max () const
- virtual void shrink (int min, int max)
- virtual bool subtract (int val)
- virtual void in_min (int min)
- virtual void in_max (int max)
- virtual void print () const

  print info about the current domain

Protected Member Functions

- **IntVariable** (int idv)

Protected Attributes

- IntDomainPtr _domain_ptr

6.28.1 Member Function Documentation

6.28.1.1 size_t IntVariable::get_size ( ) const [virtual]

It returns the size of the current domain.

Returns

the size of the current variable's domain.

Implements Variable.

6.28.1.2 void IntVariable::in_max (int *max*) [virtual]

It updates the domain according to the maximum value.

Parameters

max	domain value.

6.28.1.3 void IntVariable::in_min (int *min*) [virtual]

It updates the domain according to the minimum value.

Parameters

min	domain value.

6.28.1.4 bool IntVariable::is_empty ( ) const [virtual]

It checks if the domain is empty.

Returns

true if variable domain is empty. false otherwise.

Implements Variable.

6.28.1.5 bool IntVariable::is_singleton ( ) const [virtual]

It checks if the domain contains only one value.

Returns

true if the the variable's domain is a singleton, false otherwise.

Implements Variable.

6.28.1.6 int IntVariable::max ( ) const [virtual]

It returns the current maximal value in the domain of this variable.

Returns

the maximum value belonging to the domain.

6.28.1.7 int IntVariable::min ( ) const [virtual]

It returns the current minimal value in the domain of this variable.

Returns

the minimum value belonging to the domain.

6.28.1.8 virtual void IntVariable::set_domain ( ) [pure virtual]

Set domain's bounds. If no bounds are provided, an unbounded domain (int) is istantiated. If an array of elements A is provided, the function instantiates a domain D = [min A, max A], deleting all the elements d in D s.t. d does not belong to A.

Implemented in CudaVariable.

6.28.1.9 virtual void IntVariable::set_domain (int *lw*, int *ub*) [pure virtual]

Set domain's bounds. A new domain [lw, ub] is generated.

Parameters

lw	lower bound
ub	upper bound

Implemented in CudaVariable.

6.28.1.10 virtual void IntVariable::set_domain (std::vector< std::vector< int > > *elems*) [pure virtual]

Set domain's elements. A domain {d_1, ..., d_n} is generated.

Parameters

elems	vector of vectors (subsets) of domain's elements

Todo implement set of sets of elements.

Implemented in CudaVariable.

6.28.1.11 void IntVariable::set_domain_type (DomainType *dt*) [virtual]

Set domain according to the specific variable implementation.

Note

: different types of variable

Parameters

dt	domain type of type DomainType to set to the current variable

Implements Variable.

6.28.1.12 void IntVariable::shrink (int *min*, int *max*) [virtual]

Set domain's bounds. It updates the domain to have values only within the interval min..max.

Note

it does not update _lower_bound and _upper_bound here for efficiency reasons.

Parameters

lower	lower bound value
upper	upper bound value

6.28.1.13 bool IntVariable::subtract (int *val*) [virtual]

It intersects with the domain which is a complement of the value given as input, i.e., subtract a value from the current domain.

Parameters

val	the value to subtract from the current domain

Returns

true if succeed, false otherwise.

6.28.2 Member Data Documentation

6.28.2.1 IntDomainPtr IntVariable::_domain_ptr [protected]

Pointer to the domain of the variable. IntDomain for IntVariable

The documentation for this class was generated from the following files:

- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/int_variable.h
- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/int_variable.cpp

6.29 Logger Class Reference

Public Member Functions

- void **set_out_file** (std::string)
- void **set_verbose** (bool)
- void message (std::string)

  Print message on stdout or file (print_message force printing)
- void **print_message** (std::string)
- void log (std::string)

  Print log on stdout or file.
- void **oflog** (std::string)
- void error (std::string)

  *Print error message on cerr (optional: **FILE** and **LINE**)*
- void **error** (std::string, const char *)
- void **error** (std::string, const char *, const int)

Static Public Member Functions

- static Logger * get_instance (std::string log_file="")

  Constructor get (static) instance.

Protected Member Functions

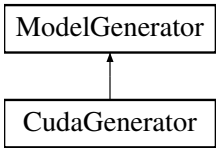
- **Logger** (std::string="")

The documentation for this class was generated from the following files:

- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/logger.h
- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/logger.cpp

6.30 ModelGenerator Class Reference

Inheritance diagram for ModelGenerator:



Public Member Functions

- virtual VariablePtr get_variable (TokenPtr)=0
- virtual ConstraintPtr get_constraint (TokenPtr)=0
- virtual SearchEnginePtr get_search_engine (TokenPtr)=0

6.30.1 Member Function Documentation

6.30.1.1 virtual ConstraintPtr ModelGenerator::get_constraint (TokenPtr) [pure virtual]

These methods create the instances of the objects and return the correspondent (shared) pointers to them.

Parameters

TokenPtr	pointer to the token describing a constraint. If the token does not correspond to the object to instantiate, it returns nullptr.

Implemented in CudaGenerator.

6.30.1.2 virtual SearchEnginePtr ModelGenerator::get_search_engine (TokenPtr) [pure virtual]

These methods create the instances of the objects and return the correspondent (shared) pointers to them.

Parameters

TokenPtr	pointer to the token describing a search engine. If the token does not correspond to the object to instantiate, it returns nullptr.

Implemented in CudaGenerator.

6.30.1.3 virtual VariablePtr ModelGenerator::get_variable (TokenPtr) [pure virtual]

These methods create the instances of the objects and return the correspondent (shared) pointers to them.

Parameters

TokenPtr	pointer to the token describing a variable. If the token does not correspond to the object to instantiate, it returns nullptr.

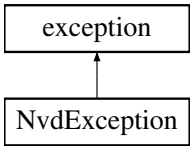
Implemented in CudaGenerator.

The documentation for this class was generated from the following file:

- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/model_generator.h

6.31 NvdException Class Reference

Inheritance diagram for NvdException:



Public Member Functions

- NvdException (const char *msg="")
- NvdException (const char *msg, const char *file)
- NvdException (const char *msg, const char *file, int line)
- virtual const char * what () const noexcept

Protected Attributes

- int _expt_line

  Code line where the exception was thrown.
- std::string _expt_file

  Name of the file where the exception was thrown.
- std::string _expt_message

  Exception message.

6.31.1 Constructor & Destructor Documentation

6.31.1.1 NvdException::NvdException (const char * *msg* = " ")

Constructor.

Parameters

msg	the message related to the exception.

6.31.1.2 NvdException::NvdException (const char * *msg*, const char * *file*)

Constructor.

Parameters

msg	the message related to the exception.
file	where the excpetion has been raised.

6.31.1.3 NvdException::NvdException (const char * *msg*, const char * *file*, int *line*)

Constructor.

Parameters

msg	the message related to the exception.
file	where the excpetion has been raised.
line	of code where the excpetion has been raised.

6.31.2 Member Function Documentation

6.31.2.1 const char * NvdException::what ( ) const [virtual], [noexcept]

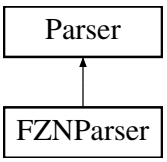
Overwrite the what method to print other information about the exception.

The documentation for this class was generated from the following files:

- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/nvd_exception.h
- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/nvd_exception.cpp

6.32 Parser Class Reference

Inheritance diagram for Parser:



Public Member Functions

- void set_input (std::string)

  Set input.
- void add_delimiter (std::string)

  Add delimiter to tokenizer.
- int get_current_line ()

  Get current (parsed) line.
- bool is_failed () const

  Check whether the parser has failed.
- virtual bool more_tokens ()
- virtual void open ()
- virtual void close ()
- virtual std::string get_next_token ()
- virtual bool more_variables () const =0
- virtual bool **more_constraints** () const =0
- virtual bool **more_search_engines** () const =0
- virtual TokenPtr get_variable ()=0
- virtual TokenPtr **get_constraint** ()=0
- virtual TokenPtr **get_search_engine** ()=0
- virtual TokenPtr get_next_content ()=0
- virtual void print () const =0

  Print info.

Protected Member Functions

- Parser ()

  Constructor.
- **Parser** (std::string)

Protected Attributes

- Tokenization * _tokenizer

  Tokenizer: it tokenizes lines read from the input file.
- std::ifstream * _if_stream

  Input stream (from file)
- std::string **_input_path**
- std::string **_dbg**
- bool **_open_file**
- bool **_open_first_time**
- bool **_more_tokens**

- bool **_new_line**
- bool **_failure**
- int _current_line

  Number of lines read so far.
- std::string _delimiters

  Delimiter to use to tokenize words.
- std::streampos _curr_pos

  Other variables needed to move into the file.
- std::map< size_t, TokenPtr > _map_tokens

  Pointers to all tokens parsed so far.

6.32.1 Member Function Documentation

6.32.1.1 void Parser::close ( ) [virtual]

Close the file.

Note

: alternating open() and close() the client can decided how much text has to be parsed.

6.32.1.2 virtual TokenPtr Parser::get_next_content ( ) [pure virtual]

Give next Token. A Token is built from a (string) token and represents a semantic object read from the FlatZinc model given in input. It holds other useful info related to the (string) token itself, e.g., line where the token has been found. If this function is call and no other Token is available it returns nullprt.

Implemented in FZNParser.

6.32.1.3 std::string Parser::get_next_token ( ) [virtual]

Get next token. This function returns a string corresponding to the token parsed according to the internal state of the object (i.e., pointer in the text file).

6.32.1.4 virtual TokenPtr Parser::get_variable ( ) [pure virtual]

Get methods: get variables, constraints, and the search engine. They increment the counter of available tokens. The tokens are returned in order w.r.t. their variables.

Implemented in FZNParser.

6.32.1.5 bool Parser::more_tokens ( ) [virtual]

Check if the internal status has more tokens to give back to the client.

6.32.1.6 virtual bool Parser::more_variables ( ) const [pure virtual]

Get methods: more tokens of the same related type (i.e., variables, constraints, and search engine). These methods should be used together with the "get" methods.

Implemented in FZNParser.

6.32.1.7 void Parser::open ( ) [virtual]

Open the file. The file is open (if not already open) and the pointer is placed on the last position read. If the file is open for the first time, the pointer is placed on the first position.

The documentation for this class was generated from the following files:

- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/parser.h
- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/parser.cpp

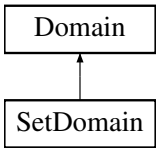
6.33 SearchEngine Class Reference

The documentation for this class was generated from the following files:

- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/search_engine.h
- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/search_engine.cpp

6.34 SetDomain Class Reference

Inheritance diagram for SetDomain:



Public Member Functions

- virtual void set_values (std::vector< int > elems)
- virtual std::vector< int > get_values () const
- DomainPtr clone () const

  Clone the current domain and returns a pointer to it.
- EventType get_event () const
- size_t get_size () const

  Returns the size of the domain.
- bool is_empty () const

  Returns true if the domain is empty.
- bool is_singleton () const

  Returns true if the domain has only one element.
- void print () const

  Print info about the domain.

Protected Member Functions

- DomainPtr **clone_impl** () const

Protected Attributes

- std::vector< int > **_d_elements**

Additional Inherited Members

6.34.1 Member Function Documentation

6.34.1.1 EventType SetDomain::get_event ( ) const [virtual]

Get event on this domain

Todo implement this function

Implements Domain.

6.34.1.2 std::vector< int > SetDomain::get_values ( ) const [virtual]

Get a vector containing the current values contained in the domain.

Returns

the current elements in the domain

6.34.1.3 void SetDomain::set_values (std::vector< int > *elems*) [virtual]

Set bounds and perform some consistency checking. It throws "no solutions" if consistency checking fails.

Parameters

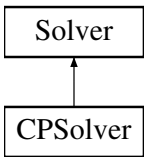
elems	vector of domain's elements

The documentation for this class was generated from the following files:

- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/set_domain.h
- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/set_domain.cpp

6.35 Solver Class Reference

Inheritance diagram for Solver:



Public Member Functions

- virtual void **run** ()=0

The documentation for this class was generated from the following file:

- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/solver.h

6.36 Statistics Class Reference

Public Member Functions

- void set_timer ()

  Set timer (starts "watching" the running time)
- void stopwatch (int tt=T_GENERAL)
- void stopwatch_and_add (int tt=T_GENERAL)
- double get_timer (int tt=T_GENERAL)
- virtual void print () const

  Print info about statistics on the program.

Static Public Member Functions

- static Statistics * get_instance ()

  Get (static) instance (singleton) of Statistics.

Static Public Attributes

- static constexpr int **T_GENERAL** = 0
- static constexpr int **T_SEARCH** = 1
- static constexpr int **T_FIRST_SOL** = 2
- static constexpr int **T_PREPROCESS** = 3
- static constexpr int **T_FILTERING** = 4

Protected Attributes

- std::string _dbg

  Debug string info.
- timeval **_time_stats**
- double **_time_start**
- double **_time** [MAX_T_TYPE]

Static Protected Attributes

- static constexpr double USEC = 1000000.0

  USEC unit.
- static constexpr int MAX_T_TYPE = 10

  Max size of the array of times.

6.36.1 Member Function Documentation

6.36.1.1 double Statistics::get_timer (int *tt* = `T_GENERAL`)

Get the value of the running time in seconds.

Parameters

tt	describes which kind of computation time must be returned,

Returns

the computational time related to tt in seconds.

6.36.1.2 void Statistics::stopwatch (int *tt* = T_GENERAL)

Stop watching the running time.

Parameters

tt	describes which kind of computation has been observed

6.36.1.3 void Statistics::stopwatch_and_add (int *tt* = T_GENERAL)

Stop watching the running time and add the time to the previous times watched for tt.

Parameters

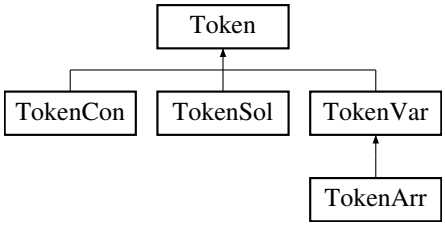
tt	describes which kind of computation has been observed

The documentation for this class was generated from the following files:

- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/statistics.h
- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/statistics.cpp

6.37 Token Class Reference

Inheritance diagram for Token:



Public Member Functions

- **Token** (TokenType)
- int **get_id** () const
- void **set_type** (TokenType)
- TokenType **get_type** () const
- virtual void print () const

  Print info about the token.

Protected Attributes

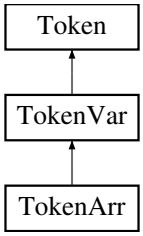
- std::string **_dbg**
- TokenType **_tkn_type**

The documentation for this class was generated from the following files:

- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/token.h
- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/token.cpp

6.38 TokenArr Class Reference

Inheritance diagram for TokenArr:



Public Member Functions

- void **set_size_arr** (int)
- int **get_size_arr** () const
- void set_array_bounds (int lw, int up)
- int **get_lw_bound** () const
- int **get_up_bound** () const
- int get_lower_var () const
- int get_upper_var () const
- bool is_var_in (int var) const
- bool **is_var_in** (std::string) const
- void set_output_arr ()

  Identifies the current variable array as a support variable array.
- bool **is_output_arr** () const
- void print () const

  Print info methods.

Additional Inherited Members

6.38.1 Member Function Documentation

6.38.1.1 int TokenArr::get_lower_var ( ) const

Variables (idx) within the array. The index is given w.r.t. the global index of parsed tokens so far.

Returns

the lower idx of variable within the array

6.38.1.2 int TokenArr::get_upper_var ( ) const

Variables (idx) within the array. The index is given w.r.t. the global index of parsed tokens so far.

Returns

the higher idx of variable within the array

6.38.1.3 bool TokenArr::is_var_in (int *var*) const

Check whether a given variable (idx) is indexed by the array (i.e., is whithin the array.

Note

: check is performed w.r.t. both the variable string identifier (e.g., a[i]) and its global id.

Parameters

var	the variable to check membership

Returns

true if var is in the current array, false otherwise

6.38.1.4 void TokenArr::set_array_bounds (int *lw,* int *up*)

Array set and info. For example, array [1..30] of ... get_lw_bound -> 1 get_lw_bound -> 30 It sets the bounds of the array.

Parameters

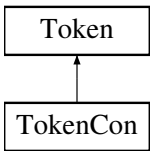
lw	lower bound
up	upper bound

The documentation for this class was generated from the following files:

- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/token_arr.h
- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/token_arr.cpp

6.39 TokenCon Class Reference

Inheritance diagram for TokenCon:



Public Member Functions

- void set_con_id (std::string)

  Set method constraint id (i.e., constraint's name).
- std::string get_con_id () const

  Get the string representing the constraint's name.
- void add_expr (std::string str)
- int get_num_expr () const

  Get the number of parameters needed by the constraint.
- std::string get_expr (int) const
- const std::vector< std::string > get_expr_array ()
- const std::vector< std::string > get_expr_elements_array ()
- const std::vector< std::string > get_expr_var_elements_array ()

- const std::vector< std::string > get_expr_not_var_elements_array ()
- virtual void print () const

  Print info methods.

Protected Attributes

- std::string _con_id

  Info about the constraint.
- std::vector< std::string > _exprs

  Parameters involved in the constraint.

6.39.1 Member Function Documentation

6.39.1.1 void TokenCon::add_expr (std::string *str*)

Add expression (parameters) to the token that identifies the parsed constraint. For example, constraint int_ne(magic[1], magic[2]) expression = "magic[1]" and "magic[2]"

Parameters

str	string representing the expression.

6.39.1.2 std::string TokenCon::get_expr (int *idx*) const

Get the string represeting the ith expression that defines the constraint.

Parameters

idx	index of the expression to return

Returns

return the idx^th expression

6.39.1.3 const std::vector< std::string > TokenCon::get_expr_array ( )

Return an array containing all the (string) expressions that define the current constraint.

Returns

a vector of strings representing the expressions defining this constraint.

6.39.1.4 const std::vector< std::string > TokenCon::get_expr_elements_array ( )

Return an array containing all the (string) elements of each expression that define the current constraint.

Returns

a vector of strings representing the elements of each expression that defines this constraint.

Note

the strings in output preserves the order as found in the original string token.

6.39.1.5 const std::vector< std::string > TokenCon::get_expr_not_var_elements_array ( )

Return an array containing all the (string) "non variable" elements of each expression that define the current constraint.

Returns

a vector of strings representing the "non variable" elements of each expression that defines this constraint.

Note

the strings in output preserves the order as found in the original string token.

6.39.1.6 const std::vector< std::string > TokenCon::get_expr_var_elements_array ( )

Return an array containing all the (string) "variable" elements of each expression that define the current constraint.

Returns

a vector of strings representing the "variable" elements of each expression that defines this constraint.

Note

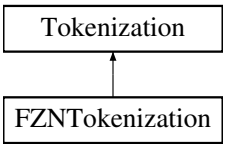
the strings in output preserves the order as found in the original string token.

The documentation for this class was generated from the following files:

- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/token_con.h
- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/token_con.cpp

6.40 Tokenization Class Reference

Inheritance diagram for Tokenization:



Public Member Functions

- void **add_delimiter** (std::string)
- void **set_delimiter** (std::string)
- void **add_white_spaces** (std::string)
- void **set_white_spaces** (std::string)
- void set_new_tokenizer (std::string line)
- bool find_new_line ()

  Informs whether a new line has been found.
- bool is_failed () const

  Check whether the tokenizer has failed.
- bool need_line ()

Asks whether the tokenizer has finished all the tokens.

- void add_comment_symb (char)

  Set preferences.
- void **add_comment_symb** (std::string)
- virtual TokenPtr get_token ()=0

  Get the string correspondent to the (filtered) token.

Protected Member Functions

- virtual bool avoid_char (char)

  It states whether the current char has to be skipped or not.
- virtual bool skip_line ()

  It states whether_c_token or the a line have to be skipped or not.
- virtual bool **skip_line** (std::string)
- virtual bool set_new_line ()
- virtual void clear_line ()
- virtual TokenPtr analyze_token ()=0

Protected Attributes

- std::string **_dbg**
- std::string **DELIMITERS** = "\t\r\n "
- std::string **WHITESPACE** = " \t"
- std::string **_comment_lines**
- bool **_new_line**
- bool **_need_line**
- bool **_failed**
- char * _c_token

  Token returned by strtok.
- char * _parsed_line

  Parsed line.

6.40.1 Member Function Documentation

6.40.1.1 virtual TokenPtr Tokenization::analyze_token ( ) [protected], [pure virtual]

Analyze token: this function acts like a filter. It analyzes _c_token and returns a string corresponding to the token cleaned from useless chars.

6.40.1.2 void Tokenization::clear_line ( ) [protected], [virtual]

It "clears" the text line by removing possible initial white spaces from line. Different heuristics may be used here.

6.40.1.3 bool Tokenization::set_new_line ( ) [protected], [virtual]

It states whether a new line has been found. Different heuristics may be used here.

6.40.1.4 void Tokenization::set_new_tokenizer (std::string *line*)

Prepare a new tokenizer (i.e., string for strtok).

Parameters

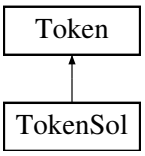
line	the string to tokenize.

The documentation for this class was generated from the following files:

- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/tokenization.h
- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/tokenization.cpp

6.41 TokenSol Class Reference

Inheritance diagram for TokenSol:



Public Member Functions

- void **set_var_goal** (std::string)
- void **set_solve_goal** (std::string)
- void **set_solve_params** (std::string)
- void **set_label_choice** (std::string)
- void **set_search_choice** (std::string)
- void **set_variable_choice** (std::string)
- void **set_assignment_choice** (std::string)
- void **set_strategy_choice** (std::string)
- void set_var_to_label (std::string)

  Set the (string) identifier of a variable to label.
- std::string **get_var_goal** () const
- std::string **get_solve_goal** () const
- std::string **get_search_choice** () const
- std::string **get_label_choice** () const
- std::string **get_variable_choice** () const
- std::string **get_assignment_choice** () const
- std::string **get_strategy_choice** () const
- int num_var_to_label () const
- const std::vector< std::string > get_var_to_label () const
- std::string get_var_to_label (int idx) const
- virtual void print () const

  Print info methods.

Protected Attributes

- std::string **_var_goal**
- std::string **_solve_goal**
- std::string **_search_choice**
- std::string **_label_choice**
- std::string **_variable_choice**
- std::string **_assignment_choice**
- std::string **_strategy_choice**
- std::vector< std::string > _var_to_label

6.41.1 Member Function Documentation

6.41.1.1 const vector< std::string > TokenSol::get_var_to_label ( ) const

Identifiers of the variables to label.

Returns

a vector of string identifiers of the variable to label during the search phase.

6.41.1.2 string TokenSol::get_var_to_label (int *idx*) const

Get the string corresponding to the ith variable to label.

Parameters

idx	the index of the variable to label.

Returns

the string identifier of the idx^th variable to label.

6.41.1.3 int TokenSol::num_var_to_label ( ) const

Number of variables to label if specified by the model.

Returns

the number of variables to label.

6.41.2 Member Data Documentation

6.41.2.1 std::vector< std::string > TokenSol::_var_to_label [protected]

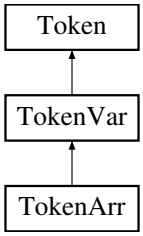
Vector of strings corresponding to the variables to label during the search phase.

The documentation for this class was generated from the following files:

- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/token_sol.h
- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/token_sol.cpp

6.42 TokenVar Class Reference

Inheritance diagram for TokenVar:



Public Member Functions

- void set_var_id (std::string str)
- std::string get_var_id () const

  Get the string id of the current variable.
- void set_objective_var ()

  Identifies the current variable as an objective variable.
- bool **is_objective_var** () const
- void set_support_var ()

  Identifies the current variable as a support variable.
- bool **is_support_var** () const
- void set_var_dom_type (VarDomainType vdt)
- VarDomainType **get_var_dom_type** () const
- void set_boolean_domain ()

  Specifies a boolean domain for the variable.
- void set_float_domain ()

  Specifies a float domain for the variable.
- void set_int_domain ()

  Specifies an integer domain for the variable.
- void set_range_domain (std::string str)
- void set_range_domain (int lw, int ub)
- int **get_lw_bound_domain** () const
- int **get_up_bound_domain** () const
- void set_subset_domain (std::string str)
- void set_subset_domain ()
- void set_subset_domain (const std::vector< int > &elems)
- void set_subset_domain (const std::vector< std::vector< int > > &elems)
- void set_subset_domain (const std::pair< int, int > &range)
- const std::vector< std::vector < int > > get_subset_domain ()
- virtual void print () const

  Print info methods.

Protected Member Functions

- std::pair< int, int > get_range (std::string str) const
- std::vector< int > get_subset (std::string str) const

Protected Attributes

- std::string **_var_id**
- bool **_objective_var**
- bool **_support_var**
- VarDomainType **_var_dom_type**
- int **_lw_bound**
- int **_up_bound**
- std::vector< std::vector< int > > **_subset_domain**

6.42.1 Member Function Documentation

6.42.1.1 pair< int, int > TokenVar::get_range (std::string *str*) const [protected]

Get a pair <x1, x2> from a string of type "∗x1..x2∗".

Parameters

str	string to parse

Returns

a pair representing the range expressed with str

6.42.1.2 vector< int > TokenVar::get_subset (std::string *str*) const [protected]

Get a vector of elements from a string of type "∗{x1, x2, ...xk}∗".

Parameters

str	string to parse

Returns

a pair representing the range expressed with str

6.42.1.3 const vector< vector< int > > TokenVar::get_subset_domain ( )

Get the set of subsets of values for a var set type.

Returns

a vector of vectors of values representing the subsets of the var set type domain.

6.42.1.4 void TokenVar::set_range_domain (std::string *str*)

Specifies a range domain for the variable with a given a string of type "∗x1..x2∗".

6.42.1.5 void TokenVar::set_range_domain (int *lw,* int *ub*)

Specifies a range domain for the variable with a given lower and upper bound.

Parameters

lw	lower bound
ub	upper bound

6.42.1.6 void TokenVar::set_subset_domain (std::string *str*)

Call the right subset function, parsing the string given in input.

6.42.1.7 void TokenVar::set_subset_domain ( )

Specifies a set of int domain.

Note

set of int;

6.42.1.8 void TokenVar::set_subset_domain (const std::vector< int > & *elems*)

Specifies a subsets of set domain for the variable with the given vector of elements.

Parameters

elems	vector of elements

Note

set of {x1, x2, ...xk}

6.42.1.9 void TokenVar::set_subset_domain (const std::vector< std::vector< int > > & *elems*)

Specifies a subsets of set domain for the variable with the given vector of elements.

Parameters

elems	vector of vectors of elements

Note

set as {{x1, x2, ...xk}, ...}

6.42.1.10 void TokenVar::set_subset_domain (const std::pair< int, int > & *range*)

Specifies a set of ints in range domain for the variable with the given range.

Parameters

range	pair of int elements for range

Note

set of x1..x2

6.42.1.11 void TokenVar::set_var_dom_type (VarDomainType *vdt*)

Set the type of the current (token) variable.

Parameters

vdt	the variable domain type of type VarDomainType.

6.42.1.12 void TokenVar::set_var_id (std::string *str*)

Set the (string) identifier of the variable represented as a token. The id is retrieved using the get_var_id() method.

Parameters

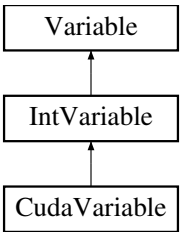
str	the string identifier of the variable.

The documentation for this class was generated from the following files:

- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/token_var.h
- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/token_var.cpp

6.43 Variable Class Reference

Inheritance diagram for Variable:



Public Member Functions

- **Variable** (int)
- int get_id () const

  Get integer id of this variable.
- void set_str_id (std::string str)
- std::string get_str_id () const
- void set_type (VariableType vt)

  Set the type of variable (i.e., FD_VARIABLE, SUP_VARIABLE, etc.)
- VariableType get_type () const

  Get the type of variable (i.e., FD_VARIABLE, SUP_VARIABLE, etc.)
- virtual const DomainPtr domain ()=0

  Get (const) reference to this domain.
- virtual EventType get_event () const =0

  Get event on this domain.
- virtual void set_domain_type (DomainType dt)=0
- virtual size_t get_size () const =0
- virtual bool is_singleton () const =0
- virtual bool is_empty () const =0
- virtual void attach_constraint (ObserverPtr c)
- virtual void detach_constraint (ObserverPtr c)
- virtual void detach_constraint (size_t c_id)
- virtual void notify_constraint ()
- virtual void notify_store ()
- virtual size_t size_constraints ()
- virtual size_t size_constraints_original () const
- virtual void print () const

  Print info about the variable.

Protected Attributes

- std::string **_dbg**
- int **_id**
- std::string **_str_id**
- VariableType **_var_type**
- size_t _number_of_observers

  Total number of observers.
- std::list< ObserverPtr > _observers
- std::list< size_t > _detach_observers

6.43.1 Member Function Documentation

6.43.1.1 void Variable::attach_constraint (ObserverPtr *c*) [virtual]

It registers constraint with this variable, so always when this variable is changed the constraint is reevaluated/notified.

Parameters

c	the (pointer to) the constraint which is added to this variable.

6.43.1.2 void Variable::detach_constraint (ObserverPtr *c*) [virtual]

It detaches constraint from this variable, so change in variable will not cause constraint reevaluation.

Parameters

c	the (pointer to) the constraint which is detached from this variable.

Note

If c appears only to be attached to this variable, this method actually destroyes the constraint c. The client must be care of storing c somewhere else in order to restore the state (e.g. for backtrack actions).

6.43.1.3 void Variable::detach_constraint (size_t *c_id*) [virtual]

It detaches constraint from this variable, so change in variable will not cause constraint reevaluation.

Parameters

c	the id of the constraint which is detached from this variable.

Note

If c appears only to be attached to this variable, this method actually destroyes the constraint c. The client must be care of storing c somewhere else in order to restore the state (e.g. for backtrack actions).

6.43.1.4 virtual size_t Variable::get_size ( ) const [pure virtual]

It returns the size of the current domain.

Returns

the size of the current variable's domain.

Implemented in IntVariable.

6.43.1.5 string Variable::get_str_id ( ) const

Get the string id of this variable.

Returns

a string representing the id of this variable.

6.43.1.6 bool Variable::is_empty ( ) const [pure virtual]

It checks if the domain is empty.

Returns

true if variable domain is empty. false otherwise.

Implemented in IntVariable.

6.43.1.7 virtual bool Variable::is_singleton ( ) const [pure virtual]

It checks if the domain contains only one value.

Returns

true if the the variable's domain is a singleton, false otherwise.

Implemented in IntVariable.

6.43.1.8 void Variable::notify_constraint ( ) [virtual]

It notifies all the constraints attached to this variables that a change has been done on this very variable.

6.43.1.9 void Variable::notify_store ( ) [virtual]

It notifies the current store attached to this variable that a change has been done on this very variable.

6.43.1.10 virtual void Variable::set_domain_type (DomainType *dt*) [pure virtual]

Set domain according to the specific variable implementation.

Note

: different types of variable

Parameters

dt	domain type of type DomainType to set to the current variable

Implemented in IntVariable.

6.43.1.11 void Variable::set_str_id (std::string *str*)

Set the (string) id of the variable.

Parameters

str	the string to set as variable's identifier

6.43.1.12 size_t Variable::size_constraints ( ) [virtual]

It returns the current number of constraints attached to this variable and that are not yet satisfied.

Returns

number of constraints attached to the variable not yet satisfied.

Note

use this method to implement some heuristics (e.g., min conflict heuristic.

6.43.1.13 size_t Variable::size_constraints_original ( ) const [virtual]

It returns the current number of constraints attached to this variable (either satisfied or not satisfied yet).

Returns

number of constraints attached to the variable.

6.43.2 Member Data Documentation

6.43.2.1 std::list<size_t> Variable::_detach_observers [protected]

List of ids of detached observers from this variable. These ids (i.e., constraints' ids) will be used to restore the variable's state during search.

Note

|_observer| + |_detach_observers| = _number_of_observers.

6.43.2.2 std::list<ObserverPtr> Variable::_observers [protected]

List of observers of this variable. These observers (i.e., constraints) will be notified when a variable is changed.

The documentation for this class was generated from the following files:

- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/variable.h
- /Users/fedecampe/Desktop/NVIDIOSO-PRJ/NVIDIOSO/NVIDIOSO/variable.cpp

Index