

NVIDIOSO

1.0

Generated by Doxygen 1.8.7

Wed Jul 16 2014 13:08:01

Contents

1	Main Page	1
2	NVIDIOSO	3
3	Todo List	5
4	Hierarchical Index	7
4.1	Class Hierarchy	7
5	Class Index	9
5.1	Class List	9
6	Class Documentation	11
6.1	BoolDomain Class Reference	11
6.1.1	Member Function Documentation	12
6.1.1.1	get_event	12
6.2	ConcreteDomain< T > Class Template Reference	12
6.2.1	Member Function Documentation	12
6.2.1.1	add	12
6.2.1.2	add	12
6.2.1.3	contains	13
6.2.1.4	get_representation	13
6.2.1.5	get_singleton	13
6.2.1.6	in_max	13
6.2.1.7	in_min	13
6.2.1.8	is_empty	14
6.2.1.9	is_singleton	14
6.2.1.10	print	14
6.2.1.11	shrink	14
6.3	Constraint Class Reference	14
6.4	ConstraintStore Class Reference	15
6.5	CPModel Class Reference	15
6.5.1	Member Function Documentation	15

6.5.1.1	add_constraint	15
6.5.1.2	add_search_engine	15
6.5.1.3	add_variable	15
6.6	CPSolver Class Reference	16
6.7	CPStore Class Reference	16
6.7.1	Member Function Documentation	17
6.7.1.1	init_model	17
6.8	CudaConcreteDomain Class Reference	17
6.8.1	Constructor & Destructor Documentation	18
6.8.1.1	CudaConcreteDomain	18
6.8.2	Member Function Documentation	18
6.8.2.1	flush_domain	18
6.8.2.2	get_alloc_bytes	18
6.8.2.3	get_num_chunks	18
6.8.2.4	is_empty	18
6.8.2.5	set_empty	18
6.9	CudaConcreteDomainBitmap Class Reference	19
6.9.1	Constructor & Destructor Documentation	19
6.9.1.1	CudaConcreteDomainBitmap	19
6.9.1.2	CudaConcreteDomainBitmap	20
6.9.2	Member Function Documentation	20
6.9.2.1	add	20
6.9.2.2	add	20
6.9.2.3	contains	20
6.9.2.4	get_representation	21
6.9.2.5	get_singleton	21
6.9.2.6	IDX_BIT	21
6.9.2.7	IDX_CHUNK	21
6.9.2.8	in_max	22
6.9.2.9	in_min	22
6.9.2.10	is_singleton	22
6.9.2.11	print	22
6.9.2.12	shrink	22
6.9.3	Member Data Documentation	23
6.9.3.1	BITS_IN_BYTE	23
6.9.3.2	BITS_IN_CHUNK	23
6.10	CudaConcreteDomainList Class Reference	23
6.10.1	Constructor & Destructor Documentation	24
6.10.1.1	CudaConcreteDomainList	24
6.10.2	Member Function Documentation	24

6.10.2.1	add	24
6.10.2.2	add	24
6.10.2.3	contains	24
6.10.2.4	find_next_pair	25
6.10.2.5	find_pair	26
6.10.2.6	find_prev_pair	26
6.10.2.7	get_representation	26
6.10.2.8	get_singleton	26
6.10.2.9	in_max	26
6.10.2.10	in_min	27
6.10.2.11	is_singleton	27
6.10.2.12	print	27
6.10.2.13	shrink	27
6.10.3	Member Data Documentation	27
6.10.3.1	_domain_size	27
6.11	CudaDomain Class Reference	28
6.11.1	Member Function Documentation	30
6.11.1.1	add_element	30
6.11.1.2	add_element_bitmap	30
6.11.1.3	add_element_bitmaplist	30
6.11.1.4	add_element_list	30
6.11.1.5	EVT_IDX	31
6.11.1.6	get_allocated_bytes	31
6.11.1.7	get_size	31
6.11.1.8	IDX_BIT	31
6.11.1.9	IDX_CHUNK	31
6.11.1.10	init_domain	31
6.11.1.11	num_chunks	32
6.11.1.12	prepare_bit_list	32
6.11.1.13	set_bounds	32
6.11.1.14	switch_list_to_bitmaplist	32
6.11.1.15	update_bitmap	33
6.11.1.16	update_bitmap_list	34
6.11.1.17	update_domain	34
6.11.1.18	update_list	34
6.11.2	Member Data Documentation	34
6.11.2.1	_domain	34
6.11.2.2	_num_allocated_bytes	34
6.11.2.3	_num_int_chunks	34
6.11.2.4	BITS_IN_BYTE	35

6.11.2.5	MAX_BYTES_SIZE	35
6.11.2.6	MAX_DOMAIN_VALUES	35
6.11.2.7	MAX_STATUS_SIZE	35
6.11.2.8	SHARED_MEM_KB	35
6.12	CudaGenerator Class Reference	35
6.13	CudaVariable Class Reference	36
6.13.1	Constructor & Destructor Documentation	36
6.13.1.1	CudaVariable	36
6.13.1.2	CudaVariable	36
6.13.2	Member Function Documentation	37
6.13.2.1	set_domain	37
6.13.2.2	set_domain	37
6.13.2.3	set_domain	37
6.14	DataStore Class Reference	37
6.14.1	Constructor & Destructor Documentation	38
6.14.1.1	DataStore	38
6.14.2	Member Function Documentation	38
6.14.2.1	load_model	38
6.15	Domain Class Reference	38
6.15.1	Member Function Documentation	39
6.15.1.1	set_type	39
6.16	FactoryModelGenerator Class Reference	40
6.17	FactoryParser Class Reference	40
6.18	FZNPaser Class Reference	40
6.18.1	Member Function Documentation	41
6.18.1.1	get_constraint	41
6.18.1.2	get_next_content	41
6.18.1.3	get_search_engine	41
6.18.1.4	get_variable	41
6.19	FZNTokenization Class Reference	41
6.19.1	Member Function Documentation	42
6.19.1.1	get_token	42
6.20	IdGenerator Class Reference	42
6.20.1	Constructor & Destructor Documentation	43
6.20.1.1	IdGenerator	43
6.21	InputData Class Reference	43
6.21.1	Constructor & Destructor Documentation	43
6.21.1.1	InputData	43
6.22	IntDomain Class Reference	43
6.22.1	Member Function Documentation	44

6.22.1.1	add_element	44
6.22.1.2	get_size	45
6.22.1.3	in_max	45
6.22.1.4	in_min	45
6.22.1.5	init_domain	45
6.22.1.6	set_bounds	45
6.22.1.7	set_singleton	46
6.22.1.8	subtract	46
6.23	Logger Class Reference	46
6.24	ModelGenerator Class Reference	47
6.24.1	Member Function Documentation	47
6.24.1.1	get_constraint	47
6.24.1.2	get_search_engine	47
6.24.1.3	get_variable	48
6.25	Parser Class Reference	48
6.25.1	Member Function Documentation	49
6.25.1.1	close	49
6.25.1.2	get_next_content	49
6.25.1.3	get_next_token	49
6.25.1.4	get_variable	50
6.25.1.5	more_tokens	50
6.25.1.6	more_variables	50
6.25.1.7	open	50
6.26	SearchEngine Class Reference	50
6.27	SetDomain Class Reference	50
6.27.1	Member Function Documentation	51
6.27.1.1	get_event	51
6.27.1.2	get_values	51
6.27.1.3	set_values	51
6.28	Solver Class Reference	52
6.29	Token Class Reference	52
6.30	TokenArr Class Reference	53
6.30.1	Member Function Documentation	53
6.30.1.1	get_lower_var	53
6.30.1.2	get_upper_var	53
6.30.1.3	is_var_in	54
6.30.1.4	set_array_bounds	54
6.31	TokenCon Class Reference	54
6.31.1	Member Function Documentation	55
6.31.1.1	add_expr	55

6.31.1.2	get_expr	55
6.31.1.3	get_expr_array	55
6.32	Tokenization Class Reference	55
6.32.1	Member Function Documentation	56
6.32.1.1	analyze_token	56
6.32.1.2	clear_line	56
6.32.1.3	set_new_line	57
6.32.1.4	set_new_tokenizer	57
6.33	TokenSol Class Reference	57
6.33.1	Member Function Documentation	58
6.33.1.1	get_var_to_label	58
6.33.1.2	get_var_to_label	58
6.33.1.3	num_var_to_label	58
6.33.2	Member Data Documentation	58
6.33.2.1	_var_to_label	58
6.34	TokenVar Class Reference	59
6.34.1	Member Function Documentation	60
6.34.1.1	get_range	60
6.34.1.2	get_subset	60
6.34.1.3	get_subset_domain	60
6.34.1.4	set_range_domain	60
6.34.1.5	set_range_domain	60
6.34.1.6	set_subset_domain	61
6.34.1.7	set_subset_domain	61
6.34.1.8	set_subset_domain	61
6.34.1.9	set_subset_domain	61
6.34.1.10	set_subset_domain	61
6.34.1.11	set_var_dom_type	61
6.34.1.12	set_var_id	62
6.35	Variable Class Reference	62
6.35.1	Member Function Documentation	62
6.35.1.1	set_domain	62
6.35.1.2	set_str_id	63
6.35.2	Member Data Documentation	63
6.35.2.1	_domain_ptr	63
Index		64

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 $|D| \leq \text{max_vector}$ or not:

- Bitmap: if $|D| \leq \text{max_vector}$;
- List of bounds: otherwise.

By default, max_vector is equal to 256. This value can be redefined via an 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 $DSZ \leq \max_vector \rightarrow REP = 0$. Moreover,
 - $\{LB, UB\}' = \{LB, k\} \{k', UB\} \rightarrow DSZ' = DSZ - (k' - k + 1)$;
 - $LB' = LB + k \rightarrow DSZ' = DSZ - (k - LB + 1)$;
 - $UB' = UB - k \rightarrow DSZ' = DSZ - (UB - k + 1)$;
- BIT: bit vector where
 - $REP < 0$: there is a total of (\leq) VECTOR_MAX bits representing REP pairs of bounds. The first part of BIT is used to store REP pairs $\langle LB, UB \rangle$. 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 $\langle LB, UB \rangle$. 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 $UB - LB + 1 \leq VECTOR_MAX$ 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 \rightarrow keep 47 kB available.

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

When a domain (or subsets of it) is (are) represented using a bitmap, the values are stored from left to right in chunks of 32 bits (considering a 32bit representation for an unsigned int), where the most significant bit is in the leftmost position of the chunk, i.e., it is the 31th bit. For example, the domain {0, 63} is store as |31...0|63...32|. The chunk is easily retrieved computing $\text{num} / 32$, while the position within each chunk can be retrieved by $\text{num} \% 32$.

Chapter 2

NVIDIOSO

NVIDIOSO - NVIDIA-based cOnstraint SOLver v. 1.0

Chapter 3

Todo List

Member `BoolDomain::get_event () const`

implement this function

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

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

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:

ConcreteDomain< T >	??
ConcreteDomain< int >	??
CudaConcreteDomain	??
CudaConcreteDomainBitmap	??
CudaConcreteDomainList	??
Constraint	??
ConstraintStore	??
CPModel	??
DataStore	??
CPStore	??
Domain	??
BoolDomain	??
IntDomain	??
CudaDomain	??
SetDomain	??
FactoryModelGenerator	??
FactoryParser	??
IdGenerator	??
InputData	??
Logger	??
ModelGenerator	??
CudaGenerator	??
Parser	??
FZNParser	??
SearchEngine	??
Solver	??
CPSolver	??
Token	??
TokenCon	??
TokenSol	??
TokenVar	??
TokenArr	??
Tokenization	??
FZNTokenization	??
Variable	??
CudaVariable	??

Chapter 5

Class Index

5.1 Class List

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

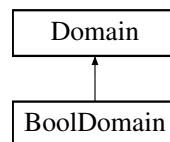
BoolDomain	??
ConcreteDomain< T >	??
Constraint	??
ConstraintStore	??
CPModel	??
CPSolver	??
CPStore	??
CudaConcreteDomain	??
CudaConcreteDomainBitmap	??
CudaConcreteDomainList	??
CudaDomain	??
CudaGenerator	??
CudaVariable	??
DataStore	??
Domain	??
FactoryModelGenerator	??
FactoryParser	??
FZNPaser	??
FZNTokenization	??
IdGenerator	??
InputData	??
IntDomain	??
Logger	??
ModelGenerator	??
Parser	??
SearchEngine	??
SetDomain	??
Solver	??
Token	??
TokenArr	??
TokenCon	??
Tokenization	??
TokenSol	??
TokenVar	??
Variable	??

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 void [shrink](#) (T min, T max)=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](#) ()=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

<i>value</i>	it specifies the value which is being added.
--------------	--

Implemented in [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

<i>min</i>	lower bound of the new domain which is being added.
------------	---

<i>max</i>	upper bound of the new domain which is being added.
------------	---

Implemented in [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

<i>value</i>	to check whether it is in the current domain.
--------------	---

Implemented in [CudaConcreteDomainBitmap](#), and [CudaConcreteDomainList](#).

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

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 invocation 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

<i>max</i>	domain value.
------------	---------------

Implemented in [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

<i>min</i>	domain value.
------------	---------------

Implemented in [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](#), 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

<i>min</i>	new lower bound to set for the current domain.
<i>max</i>	new upper bound to set for the current domain.

Implemented in [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

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 CPMModel 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 CPMModel::add_constraint (ConstraintPtr ptr) [virtual]

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

Parameters

<i>ptr</i>	pointer to the constraint to add to the model
------------	---

6.5.1.2 void CPMModel::add_search_engine (SearchEnginePtr ptr) [virtual]

Add a search engine to the model.

Parameters

<i>ptr</i>	pointer to the search engine to use to explore the search space.
------------	--

6.5.1.3 void CPMModel::add_variable (VariablePtr ptr) [virtual]

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

Parameters

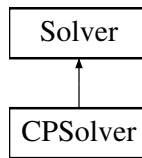
<i>ptr</i>	pointer to the variable to add to the model
------------	---

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

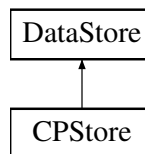
- void **run** ()

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 variables) and add them to the model. A generator takes tokens as input and returns the corresponding pointer to the instantiated objects.

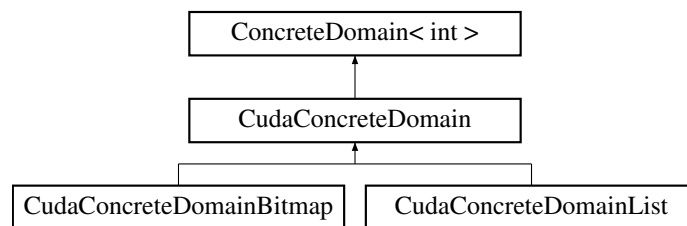
Implements [DataStore](#).

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 CudaConcreteDomain Class Reference

Inheritance diagram for CudaConcreteDomain:



Public Member Functions

- [CudaConcreteDomain](#) (size_t size)
- 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 "int" chunks allocated.
- int [_lower_bound](#)
Lower bound.
- int [_upper_bound](#)
Upper bound.
- int * [_concrete_domain](#)
Concrete domain is represented by an array of (32 bit) integers.

6.8.1 Constructor & Destructor Documentation

6.8.1.1 `CudaConcreteDomain::CudaConcreteDomain (size_t size)`

Constructor for [CudaConcreteDomain](#). It instantiates a new object and allocate size bytes for the array of integers

Parameters

<i>size</i>	the number of bytes to allocate.
-------------	----------------------------------

Note

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

6.8.2 Member Function Documentation

6.8.2.1 `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.

6.8.2.2 `size_t CudaConcreteDomain::get_alloc_bytes () const`

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

6.8.2.3 `int CudaConcreteDomain::get_num_chunks () const`

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

6.8.2.4 `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 >](#).

6.8.2.5 `void CudaConcreteDomain::set_empty () [protected]`

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

Note

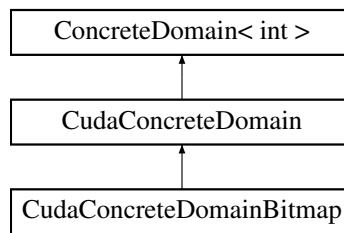
it does not flush the current internal domain's representation.

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

- `/Users/fedecampe/Desktop/NVIDIAIOSO-PRJ/NVIDIAIOSO/NVIDIAIOSO/cuda_concrete_domain.h`
- `/Users/fedecampe/Desktop/NVIDIAIOSO-PRJ/NVIDIAIOSO/NVIDIAIOSO/cuda_concrete_domain.cpp`

6.9 CudaConcreteDomainBitmap Class Reference

Inheritance diagram for CudaConcreteDomainBitmap:



Public Member Functions

- [CudaConcreteDomainBitmap](#) (size_t size)
- [CudaConcreteDomainBitmap](#) (size_t size, int min, int max)
- void [shrink](#) (int min, int max)
- 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)

Protected Attributes

- 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

6.9.1 Constructor & Destructor Documentation

6.9.1.1 CudaConcreteDomainBitmap::CudaConcreteDomainBitmap (size_t size)

Constructor for [CudaConcreteDomainBitmap](#).

Parameters

<i>size</i>	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).

6.9.1.2 `CudaConcreteDomainBitmap::CudaConcreteDomainBitmap (size_t size, int min, int max)`

Constructor for [CudaConcreteDomainBitmap](#).

Parameters

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

6.9.2 Member Function Documentation

6.9.2.1 `void CudaConcreteDomainBitmap::add (int value) [virtual]`

It computes union of this domain and {value}.

Parameters

<i>value</i>	it specifies the value which is being added.
--------------	--

Note

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

Implements [ConcreteDomain< int >](#).

6.9.2.2 `void CudaConcreteDomainBitmap::add (int min, int max) [virtual]`

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

Parameters

<i>min</i>	lower bound of the new domain which is being added.
<i>max</i>	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 >](#).

6.9.2.3 `bool CudaConcreteDomainBitmap::contains (int value) const [virtual]`

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

Parameters

<i>value</i>	to check whether it is in the current domain.
--------------	---

Note

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

Implements [ConcreteDomain< int >](#).

6.9.2.4 const void * CudaConcreteDomainBitmap::get_representation () const

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.

6.9.2.5 int CudaConcreteDomainBitmap::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 >](#).

6.9.2.6 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 chunk, i.e., the position of the corresponding bit within a chunk.

Parameters

<i>val</i>	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.

6.9.2.7 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

<i>max</i>	lower bound used to calculated the index of the bitmap
------------	--

Returns

number of int used as bitmaps to represent max

6.9.2.8 `void CudaConcreteDomainBitmap::in_max (int max)` [virtual]

It updates the domain according to max value.

Parameters

<i>max</i>	domain value.
------------	---------------

Implements [ConcreteDomain< int >](#).

6.9.2.9 `void CudaConcreteDomainBitmap::in_min (int min)` [virtual]

It updates the domain according to min value.

Parameters

<i>min</i>	domain value.
------------	---------------

Implements [ConcreteDomain< int >](#).

6.9.2.10 `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 >](#).

6.9.2.11 `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 >](#).

6.9.2.12 `void CudaConcreteDomainBitmap::shrink (int min, int max)` [virtual]

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

Parameters

<i>min</i>	new lower bound to set for the current domain.
------------	--

<i>max</i>	new upper bound to set for the current domain.
------------	--

Implements [ConcreteDomain< int >](#).

6.9.3 Member Data Documentation

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

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

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

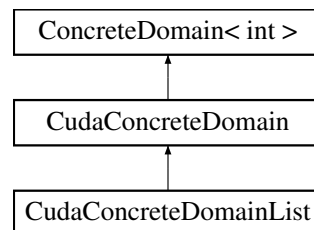
Macro for the size of a chunk in terms of bits.

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

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

6.10 CudaConcreteDomainList Class Reference

Inheritance diagram for CudaConcreteDomainList:



Public Member Functions

- [CudaConcreteDomainList](#) (size_t size, int min, int max)
- void [shrink](#) (int min, int max)
- 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.
- `int _domain_size`

6.10.1 Constructor & Destructor Documentation

6.10.1.1 `CudaConcreteDomainList::CudaConcreteDomainList (size_t size, int min, int max)`

Constructor for [CudaConcreteDomainList](#).

Parameters

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

6.10.2 Member Function Documentation

6.10.2.1 `void CudaConcreteDomainList::add (int value) [virtual]`

It computes union of this domain and {value}.

Parameters

<i>value</i>	it specifies the value which is being added.
--------------	--

Implements [ConcreteDomain< int >](#).

6.10.2.2 `void CudaConcreteDomainList::add (int min, int max) [virtual]`

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

Parameters

<i>min</i>	lower bound of the new domain which is being added.
<i>max</i>	upper bound of the new domain which is being added.

Implements [ConcreteDomain< int >](#).

6.10.2.3 `bool CudaConcreteDomainList::contains (int val) const [virtual]`

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

Parameters

<i>val</i>	to check whether it is in the current domain.
------------	---

Note

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

Implements [ConcreteDomain< int >](#).

6.10.2.4 `int CudaConcreteDomainList::find_next_pair (int val) const` `[protected]`

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

Parameters

<i>val</i>	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.

6.10.2.5 `int CudaConcreteDomainList::find_pair (int val) const` [protected]

Find the index of the pair containing *val*.

Parameters

<i>val</i>	to be searched in the list of pairs.
------------	--------------------------------------

Returns

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

6.10.2.6 `int CudaConcreteDomainList::find_prev_pair (int val) const` [protected]

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

Parameters

<i>val</i>	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.

6.10.2.7 `const void* CudaConcreteDomainList::get_representation () const`

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.

6.10.2.8 `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 >](#).

6.10.2.9 `void CudaConcreteDomainList::in_max (int max)` [virtual]

It updates the domain according to *max* value.

Parameters

<i>max</i>	domain value.
------------	---------------

Implements [ConcreteDomain< int >](#).

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

It updates the domain according to min value.

Parameters

<i>min</i>	domain value.
------------	---------------

Implements [ConcreteDomain< int >](#).

6.10.2.11 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 >](#).

6.10.2.12 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 >](#).

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

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

Parameters

<i>min</i>	new lower bound to set for the current domain.
<i>max</i>	new upper bound to set for the current domain.

Implements [ConcreteDomain< int >](#).

6.10.3 Member Data Documentation

6.10.3.1 int CudaConcreteDomainList::_domain_size [protected]

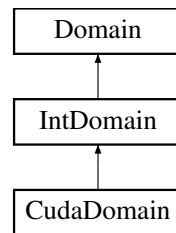
Current domain size, i.e., sum of the elements on each pair of bounds in the list.

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

6.11 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
- void [set_bounds](#) (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.
- CudaDomainRepresentation [get_representation](#) () const

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

- void `update_domain` ()
- int `update_bitmap` (int min, int max, int offset_bitmap=BIT_IDX())
- void `update_bitmap_list` ()
- void `switch_list_to_bitmaplist` ()
- void `prepare_bit_list` (int min, int max, int idx)
- void `update_list` ()
- void `add_element_bitmap` (int n)
- void `add_element_list` (int n)
- void `add_element_bitmaplist` (int n)

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

- int * `_domain`
- size_t `_num_allocated_bytes`
- size_t `_num_int_chunks`

Static Protected Attributes

- static constexpr int `INT_NO_EVT` = 0
- static constexpr int `INT_SINGLETON_EVT` = 1
- static constexpr int `INT_BOUNDS_EVT` = 2
- static constexpr int `INT_CHANGE_EVT` = 3
- static constexpr int `INT_FAIL_EVT` = 4
- static constexpr int `INT_OTHER_EVT` = 5
- 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

6.11.1 Member Function Documentation

6.11.1.1 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 maintains the current size.

Implements [IntDomain](#).

6.11.1.2 void CudaDomain::add_element_bitmap (int *n*) [protected]

Add element to a bitmap representation. It sets the corresponding bit to 1.

Parameters

<i>n</i>	the value to add.
----------	-------------------

Note

if the value is out of the current bounds the function does not add the element.
it effectively increases the size of the domain by one.

6.11.1.3 void CudaDomain::add_element_bitmaplist (int *n*) [protected]

Add element to a list representation. If the element is within a bitmap -> switch the corresponding bit to 1 (if 0) and increase size. Otherwise, add {*n*, *n*} int (i.e., 00..01) to the list of bitmaps. The new bitmap is inserted in the right position w.r.t. the elements on the right.

Parameters

<i>n</i>	the value to add.
----------	-------------------

Note

if the value is out of the current bounds the function does not add the element.
it effectively increases the size of the domain by one.
it moves all the elements greater than *n* to the right.

6.11.1.4 void CudaDomain::add_element_list (int *n*) [protected]

Add element to a list representation. It creates a new singleton {*n*, *n*} in the right position.

Parameters

<i>n</i>	the value to add.
----------	-------------------

Note

if the value is out of the current bounds the function does not add the element.
it effectively increases the size of the domain by one.
it moves all the elements greater than *n* to the right.

6.11.1.5 `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](#).

6.11.1.6 `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.

6.11.1.7 `size_t CudaDomain::get_size () const [virtual]`

Get domain size. It returns the current 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 [IntDomain](#).

6.11.1.8 `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

<i>max</i>	lower bound used to calculate the index of the bitmap
------------	---

Returns

number of int used as bitmaps to represent max

6.11.1.9 `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

<i>max</i>	lower bound used to calculate the index of the bitmap
------------	---

Returns

number of int used as bitmaps to represent max

6.11.1.10 `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 $max = MAX_INT() / 2$ and $min = MIN_INT() / 2$, etc..

Note

It instantiate an array of ints of at most MAX_BYTES_SIZE.

Parameters

<i>min</i>	lower bound of the domain
<i>max</i>	upper bound of the domain

Returns

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

Implements [IntDomain](#).

6.11.1.11 `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

<i>n</i>	number of values to represent as bits
----------	---------------------------------------

Returns

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

6.11.1.12 `void CudaDomain::prepare_bit_list (int min, int max, int idx)` `[protected]`

It sets up a pair <LB, UB> and the corresponding bitmap representation (all bits set to one).

Parameters

<i>min</i>	lower bound to store in idx
<i>max</i>	upper bound to store in idx + 1
<i>idx</i>	index position in BIT where to store LB

6.11.1.13 `void CudaDomain::set_bounds (int min, int max)` `[virtual]`

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

Parameters

<i>min</i>	lower bound
<i>max</i>	upper bound

Implements [IntDomain](#).

6.11.1.14 `void CudaDomain::switch_list_to_bitmaplist ()` `[protected]`

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

6.11.1.15 `int CudaDomain::update_bitmap (int min, int max, int offset_bitmap = BIT_IDX ())` [protected]

Update domain's bounds according to min, max and considering a bitmap representation.

Parameters

<i>min</i>	lower bound
<i>max</i>	upper bound
<i>offset_bitmap</i>	offset in BIT field where the bitmap begins. Default: BIT_IDX (). An offset of n will give a position of BIT_IDX () + n.

Returns

the number of bits set to one from min to max

6.11.1.16 void CudaDomain::update_bitmap_list () [protected]

Update domain considering bitmap list representation.

Note

Bitmap list does not change its "structure" in the BIT field. Instead, it clears the bits and states whether there is an empty domain.

6.11.1.17 void CudaDomain::update_domain () [protected]

Updates the size of the domain and the domain itself. It checks whether the domain contains "holes", e.g., {1, 2, _, 6, 7, 8} -> size = 5; Moreover, it switches from list to bitmap representation if size < VECTOR_MAX and representation is not already bitmap.

Note

this method could switch between different domain representations.

6.11.1.18 void CudaDomain::update_list () [protected]

Update domain considering a list of bounds representation. It converts into a bitmap representation if after updating size < VECTOR_MAX.

6.11.2 Member Data Documentation

6.11.2.1 int* CudaDomain::_domain [protected]

Array of int used to represent the actual domain. Operations are performed on this representation. See, [system_description.h](#)

6.11.2.2 size_t CudaDomain::_num_allocated_bytes [protected]

Total allocated bytes for representing the current domain.

6.11.2.3 size_t CudaDomain::_num_int_chunks [protected]

Total number of bitchunks.

Note

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

6.11.2.4 `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.

6.11.2.5 `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, $(48 - 1) \text{ kB} = 47 * 1024 = 48128 \text{ Byte}$.

6.11.2.6 `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 = $((\text{MAX_BYTES_SIZE} - 5 * \text{sizeof}(\text{int})) / \text{sizeof}(\text{int}))$

Note

see CUDA specifications.

6.11.2.7 `constexpr size_t CudaDomain::MAX_STATUS_SIZE = 5 * sizeof(int)` `[static], [protected]`

Number of Bytes needed for representing the current domain status.

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

Shared memory available.

Note

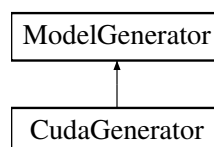
keep 1 kB less than the actual memory available.

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

6.12 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

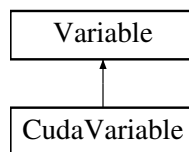
- std::string **_dbg**

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

6.13 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

6.13.1 Constructor & Destructor Documentation

6.13.1.1 CudaVariable::CudaVariable ()

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

6.13.1.2 CudaVariable::CudaVariable (int idv)

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

Parameters

<i>idv</i>	identifier to give to the variable
------------	------------------------------------

6.13.2 Member Function Documentation

6.13.2.1 void `CudaVariable::set_domain ()`

Set domain's bounds. If no bounds are provided, an unbounded domain (int) is instantiated. 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.

6.13.2.2 void `CudaVariable::set_domain (int lw, int ub)`

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

Parameters

<i>lw</i>	lower bound
<i>ub</i>	upper bound

6.13.2.3 void `CudaVariable::set_domain (std::vector< std::vector< int > > elems)`

Set domain's elements. A domain {*d*₁, ..., *d*_{*n*}} is generated.

Parameters

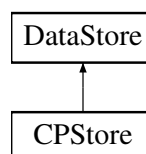
<i>elems</i>	vector of vectors (subsets) of domain's elements
--------------	--

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

6.14 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.

6.14.1 Constructor & Destructor Documentation

6.14.1.1 DataStore::DataStore (std::string in_file) [protected]

Constructor.

Parameters

<i>in_file</i>	file path of the model to parse.
----------------	----------------------------------

6.14.2 Member Function Documentation

6.14.2.1 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.

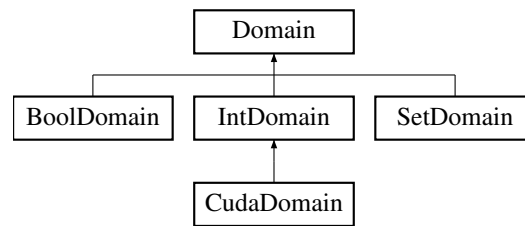
Implemented in [CPStore](#).

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

6.15 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`

6.15.1 Member Function Documentation

6.15.1.1 void Domain::set_type (DomainType dt)

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

Parameters

<code>dt</code>	domain type of type DomainType
-----------------	--------------------------------

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

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

6.16 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/NVIDIAOSO-PRJ/NVIDIAOSO/NVIDIAOSO/factory_generator.h

6.17 FactoryParser Class Reference

Static Public Member Functions

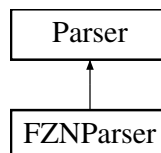
- static [Parser](#) * [get_parser](#) (ParserType pt)
Get the right parser based on the input.

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

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

6.18 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 constraints 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

6.18.1 Member Function Documentation

6.18.1.1 TokenPtr FZNPaser::get_constraint () [virtual]

Get a "constraint" token.

Returns

token pointer to a "constraint" token.

Implements [Parser](#).

6.18.1.2 TokenPtr FZNPaser::get_next_content () [virtual]

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

Set position on file to the most recent position

Implements [Parser](#).

6.18.1.3 TokenPtr FZNPaser::get_search_engine () [virtual]

Get a "search_engine" token.

Returns

token pointer to a "search_engine" token.

Implements [Parser](#).

6.18.1.4 TokenPtr FZNPaser::get_variable () [virtual]

Get a "variable" token.

Returns

token pointer to a "variable" token.

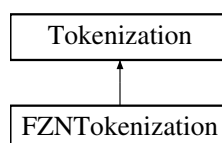
Implements [Parser](#).

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

6.19 FZNTokenization Class Reference

Inheritance diagram for FZNTokenization:



Public Member Functions

- TokenPtr [get_token](#) ()

Additional Inherited Members

6.19.1 Member Function Documentation

6.19.1.1 TokenPtr FZNTokenization::get_token () [virtual]

Specialized method: It actually gets the right token according to the FlatZinc format. Analysis is performed 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

6.20 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.

6.20.1 Constructor & Destructor Documentation

6.20.1.1 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

6.21 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[])

6.21.1 Constructor & Destructor Documentation

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

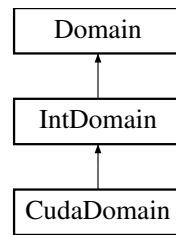
Protected constructor: a client cannot instantiate Singleton directly. Exit if the user did not set an input file!

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

6.22 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 int [get_lower_bound](#) () const
Get the domain's lower bound.
- virtual int [get_upper_bound](#) () const
Get the domain's upper bound.
- virtual void [print](#) () const
Print base info about int domain.
- virtual void [init_domain](#) (int min, int max)=0
- virtual void [set_bounds](#) (int min, int max)=0
- virtual size_t [get_size](#) () const =0
- virtual EventType [get_event](#) () const =0
Get the current event on the domain.
- 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

Protected Attributes

- int [_lower_bound](#)
- int [_upper_bound](#)

Additional Inherited Members

6.22.1 Member Function Documentation

6.22.1.1 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

<i>val</i>	element to add to the current domain.
------------	---------------------------------------

Implemented in [CudaDomain](#).

6.22.1.2 virtual size_t IntDomain::get_size () const [pure virtual]

Returns the size of the domain. This function should be implemented by derived classes according to their internal domain representation.

Note

`upper_bound - lower_bound + 1` could not be the actual size of the domain.

Returns

the current domain's size.

Implements [Domain](#).

Implemented in [CudaDomain](#).

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

It updates the domain according to the maximum value.

Parameters

<i>max</i>	domain value.
------------	---------------

Implemented in [CudaDomain](#).

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

It updates the domain according to the minimum value.

Parameters

<i>min</i>	domain value.
------------	---------------

Implemented in [CudaDomain](#).

6.22.1.5 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](#).

6.22.1.6 virtual void IntDomain::set_bounds (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

<i>lower</i>	lower bound value
<i>upper</i>	upper bound value

Implemented in [CudaDomain](#).

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

Set domain to the singleton element given in input.

Parameters

<i>val</i>	the value to set as singleton
------------	-------------------------------

Returns

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

Implemented in [CudaDomain](#).

6.22.1.8 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

<i>val</i>	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

6.23 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

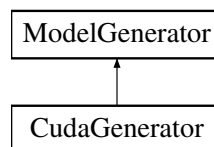
- **Logger** (std::string="")

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

6.24 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

6.24.1 Member Function Documentation

6.24.1.1 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

<i>TokenPtr</i>	pointer to the token describing a constraint. If the token does not correspond to the object to instantiate, it returns nullptr.
-----------------	--

Implemented in [CudaGenerator](#).

6.24.1.2 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

<i>TokenPtr</i>	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](#).

6.24.1.3 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

<i>TokenPtr</i>	pointer to the token describing a variable. If the token does not correspond to the object to instantiate, it returns nullptr.
-----------------	--

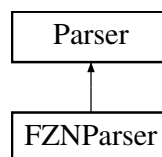
Implemented in [CudaGenerator](#).

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

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

6.25 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.

6.25.1 Member Function Documentation

6.25.1.1 void Parser::close () [virtual]

Close the file.

Note

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

6.25.1.2 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 nullptr.

Implemented in [FZNParser](#).

6.25.1.3 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).

6.25.1.4 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](#).

6.25.1.5 bool Parser::more_tokens () [virtual]

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

6.25.1.6 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](#).

6.25.1.7 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

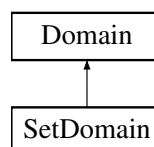
6.26 SearchEngine Class Reference

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

6.27 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

6.27.1 Member Function Documentation

6.27.1.1 `EventType SetDomain::get_event () const` `[virtual]`

Get event on this domain

Todo implement this function

Implements [Domain](#).

6.27.1.2 `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

6.27.1.3 `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

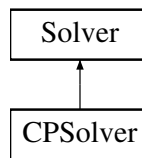
<i>elems</i>	vector of domain's elements
--------------	-----------------------------

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`

6.28 Solver Class Reference

Inheritance diagram for Solver:



Public Member Functions

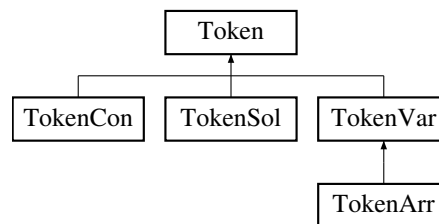
- virtual void **run** ()=0

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

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

6.29 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

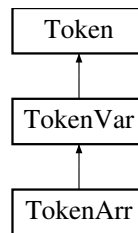
- std::string **_dbg**
- TokenType **_tkn_type**

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

6.30 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

6.30.1 Member Function Documentation

6.30.1.1 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

6.30.1.2 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

6.30.1.3 bool TokenArr::is_var_in (int var) const

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

Note

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

Parameters

<i>var</i>	the variable to check membership
------------	----------------------------------

Returns

true if var is in the current array, false otherwise

6.30.1.4 void TokenArr::set_array_bounds (int lw, int up)

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

Parameters

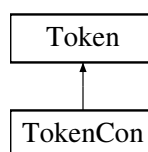
<i>lw</i>	lower bound
<i>up</i>	upper bound

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

6.31 TokenCon Class Reference

Inheritance diagram for TokenCon:



Public Member Functions

- void [set_con_id](#) (std::string)
Get/set methods.
- std::string [get_con_id](#) () const
- 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](#) ()
- 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.

6.31.1 Member Function Documentation

6.31.1.1 `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

<i>str</i>	string representing the expression.
------------	-------------------------------------

6.31.1.2 `std::string TokenCon::get_expr (int idx) const`

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

Parameters

<i>idx</i>	index of the expression to return
------------	-----------------------------------

Returns

return the idx^{th} expression

6.31.1.3 `const std::vector< std::string > TokenCon::get_expr_array ()`

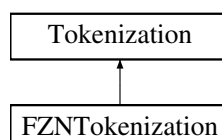
Return an array containing all the (string) expressions that define the current constraint.

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`

6.32 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** ()
Informes 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.

6.32.1 Member Function Documentation

6.32.1.1 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.

6.32.1.2 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.

6.32.1.3 `bool Tokenization::set_new_line ()` `[protected]`, `[virtual]`

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

6.32.1.4 `void Tokenization::set_new_tokenizer (std::string line)`

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

Parameters

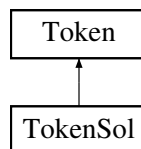
<i>line</i>	the string to tokenize.
-------------	-------------------------

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

6.33 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`

6.33.1 Member Function Documentation

6.33.1.1 `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.

6.33.1.2 `string TokenSol::get_var_to_label (int idx) const`

Get the string corresponding to the *ith* variable to label.

Parameters

<i>idx</i>	the index of the variable to label.
------------	-------------------------------------

Returns

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

6.33.1.3 `int TokenSol::num_var_to_label () const`

Number of variables to label if specified by the model.

Returns

the number of variables to label.

6.33.2 Member Data Documentation

6.33.2.1 `std::vector< std::string > TokenSol::_var_to_label` `[protected]`

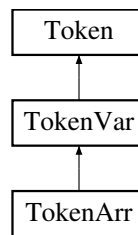
Vector of strings corresponding to the variables to label during the search phase.

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`

6.34 TokenVar Class Reference

Inheritance diagram for TokenVar:



Public Member Functions

- void [set_var_id](#) (std::string str)
- std::string [get_var_id](#) () const
- 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**

6.34.1 Member Function Documentation

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

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

Parameters

<i>str</i>	string to parse
------------	-----------------

Returns

a pair representing the range expressed with *str*

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

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

Parameters

<i>str</i>	string to parse
------------	-----------------

Returns

a pair representing the range expressed with *str*

6.34.1.3 `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.

6.34.1.4 `void TokenVar::set_range_domain (std::string str)`

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

6.34.1.5 `void TokenVar::set_range_domain (int lw, int ub)`

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

Parameters

<i>lw</i>	lower bound
<i>ub</i>	upper bound

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

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

6.34.1.7 void TokenVar::set_subset_domain ()

Specifies a set of int domain.

Note

set of int;

6.34.1.8 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

<i>elems</i>	vector of elements
--------------	--------------------

Note

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

6.34.1.9 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

<i>elems</i>	vector of vectors of elements
--------------	-------------------------------

Note

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

6.34.1.10 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

<i>range</i>	pair of int elements for range
--------------	--------------------------------

Note

set of x1..x2

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

Set the type of the current (token) variable.

Parameters

<i>vdT</i>	the variable domain type of type VarDomainType.
------------	---

6.34.1.12 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

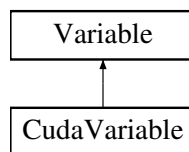
<i>str</i>	the string identifier of the variable.
------------	--

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

6.35 Variable Class Reference

Inheritance diagram for Variable:



Public Member Functions

- **Variable** (int)
- int **get_id** () const
- void **set_str_id** (std::string str)
- std::string **get_str_id** () const
- void **set_type** (VariableType vt)
- VariableType **get_type** () const
- virtual void **set_domain** (DomainType dt)
- virtual void **print** () const =0

Print info about the variable.

Protected Attributes

- std::string **_dbg**
- int **_id**
- std::string **_str_id**
- VariableType **_var_type**
- DomainPtr **_domain_ptr**

6.35.1 Member Function Documentation

6.35.1.1 void Variable::set_domain (DomainType dt) [virtual]

Set domain according to the specific variable implementation.

Note

: different types of variable

Parameters

<i>dt</i>	domain type of type DomainType to set to the current variable
-----------	---

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

Set the (string) id of the variable.

Parameters

<i>str</i>	the string to set as variable's identifier
------------	--

6.35.2 Member Data Documentation**6.35.2.1 DomainPtr Variable::_domain_ptr** [protected]

Pointer to the domain of the variable.

Note

: each variable is associated with a Finite [Domain](#).

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

- close
 - Parser, [49](#)
- Constraint, [14](#)
- Domain, [38](#)
- Logger, [46](#)
- open
 - Parser, [50](#)
- Parser, [48](#)
 - close, [49](#)
 - open, [50](#)
- Solver, [52](#)
- Token, [52](#)
- Tokenization, [55](#)
- Variable, [62](#)