Contents

1 Itk 2

1.1 FrameSexp — *A Flexible Frame Expression* 2

1.1.1 Check Special Forms 3

1.1.3 Scan 4

1.1.4 Access to Slot and Value 5

1.1.5 Duplicate a Frame 6

1.1.6 Formated Output 7

1.1.7 Heap — *A Heap of FrameSexp* 7

1.1.7.1 New Frame ..... ??

1.1.7.3 Scan 8

1.1.8 Scanner — *A Scanner of FrameSexp* 9

1.1.8.1 constructors ..... ??

1.1.9 Usage 10

Class Graph 15

1 namespace **Itk**

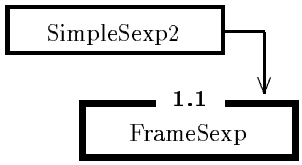
Names

1.1 class **FrameSexp** : public SimpleSexp2
A Flexible Frame Expression .. 2

1.1 class **FrameSexp** : public SimpleSexp2

A Flexible Frame Expression

Inheritance



Public Members

1.1.1 **Check Special Forms** 3
1.1.2 Bool **chkSyntax** (Int depth = -1) const
Syntax Check 3
1.1.3 **Scan** 4
1.1.4 **Access to Slot and Value** 5
1.1.5 **Duplicate a Frame** 6
1.1.6 **Formated Output** 7
1.1.7 class **Heap** : public SimpleSexp2::Heap
A Heap of FrameSexp 7
1.1.8 class **Scanner** : public SimpleSexp2::Scanner
A Scanner of FrameSexp 9
1.1.9 **Usage** 10

A Flexible Frame Expression

1.1.1 Check Special Forms

Names

Bool	**isFrame** () const	*check the data is a frame (not check in detail)*
Bool	**isSetFrame** () const	*check the data is a* set *frame or not*
Bool	**isListFrame** () const	*check the data is a* list *frame or not*
Bool	**isNotFrame** () const	*check the data is a* not *frame or not*
Bool	**isWhFrame** () const	*check the data is a* wh *frame or not*
Bool	**isNormalFrame** () const	*check the data is a normal frame*

1.1.2 Bool **chkSyntax** (Int depth = -1) const

Syntax Check

Syntax Check

If depth > 0, then check upper than the depth. If depth < 0, it checks whole frames. If depth < 0 and too deep frame (more than 2^32), then it stops.

1.1.3 Scan

Names

static FrameSexp*
scan (Scanner * scanner, Bool resetp = True,
Heap * heap = ITK_NULLPTR)
scan a frame from a scanner

static FrameSexp*
scan (Scanner & scanner, Bool resetp = True,
Heap * heap = ITK_NULLPTR)
scan a frame from a scanner

static FrameSexp*
scan (istream * istr, Bool resetp = True,
Heap * heap = ITK_NULLPTR)
scan a frame from an istream

static FrameSexp*
scan (istream & istr, Bool resetp = True,
Heap * heap = ITK_NULLPTR)
scan a frame from an istream

static FrameSexp*
scan (Buffer & buf, Bool resetp = True,
Heap * heap = ITK_NULLPTR)
scan a frame from a buffer (string)

static FrameSexp*
scan (Buffer * buf, Bool resetp = True,
Heap * heap = ITK_NULLPTR)
scan a frame from a buffer (string)

static FrameSexp*
scan (const char * str, Bool resetp = True,
Heap * heap = ITK_NULLPTR)
scan a frame from a c-string

scan() facilities scans a frame from a Scanner, istrstream, Buffer, or c-string. scan() with istrstream, Buffer, or c-string use a shard scanner of SimpleSexp2 class. This means that users need to take care of using them under multi-threading.

1.1.4 Access to Slot and Value

Names

FrameSexp*
head () *get a frame head*

FrameSexp*
setHead (SimpleSexp2 * head)
set a frame head

FrameSexp*
slotvalue (const SubString & slotname)
get a value of a slot

FrameSexp*
slotvalue (const SimpleSexp2 * slotname)
get a value of a slot

FrameSexp*
setSlot (const SubString & slotname,
SimpleSexp2 * value, Heap * heap)
set a value to a slot

FrameSexp*
setSlot (SimpleSexp2 * slotname,
SimpleSexp2 * value, Heap * heap)
set a value to a slot

Int **slotN** () const *return number of slot-value pairs*

FrameSexp*
nthSlot (Int n) *access to nth slot-value pair*

FrameSexp*
nthSlotName (Int n)
access to nth slot name

FrameSexp*
nthSlotValue (Int n)
access to nth slot value

FrameSexp*

setNthSlotName (Int n, SimpleSexp2 * slot, Heap * heap)
set nth slot

FrameSexp*
setNthSlotValue (Int n, SimpleSexp2 * value, Heap * heap)
set nth value

FrameSexp*
setNthSlot (Int n, SimpleSexp2 * slot, SimpleSexp2 * value, Heap * heap)
set nth value

Int **argN** () const *return number of args of list and set frame*

FrameSexp*
nthArg (Int n) *access to nth arg of list and set frame*

FrameSexp*
setNthArg (Int n, SimpleSexp2 *arg, Heap * heap)
set nth arg of list and set frame

FrameSexp*
content () *access to content of not and wh frame*

FrameSexp*
setContent (SimpleSexp2 * content, Heap * heap)
set content of not and wh frame

1.1.5 Duplicate a Frame

Names

FrameSexp*
dup (Heap & heap, Bool strcopyp = True) const
diplicate a frame

FrameSexp*
dup (Heap * heap, Bool strcopyp = True) const
diplicate a frame

1.1.6 Formated Output

Names

void **outputLaTeX** (ostream & ostr, Int indent = 0)
output in LaTeX format

void **outputLaTeX** (ostream * ostr, Int indent = 0)
output in LaTeX format

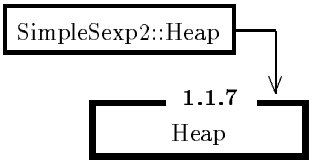
void **outputHTML** (ostream & ostr, Int indent = 0)
output in HTML format

void **outputHTMl** (ostream * ostr, Int indent = 0)
output in HTML format

1.1.7 class **Heap** : public SimpleSexp2::Heap

A Heap of FrameSexp

Inheritance



Public Members

FrameSexp*

addNewSlotValue (SimpleSexp2 * frame,
SimpleSexp2 * slot,
SimpleSexp2 * value,
Bool lastp = True)
add new slot to a frame

1.1.7.2 FrameSexp*
dup (const FrameSexp * original,
Bool strcopyp = True)
Duplicate a Frame 8

1.1.7.3 **Scan** 8

A Heap of FrameSexp

1.1.7.2

FrameSexp* **dup** (const FrameSexp * original, Bool strcopyp = True)

Duplicate a Frame

Duplicate a Frame

1.1.7.3

Scan

Names

FrameSexp*
scan (istream * istr, Bool clearheapp = False)
scan a frame from a stream

FrameSexp*
scan (istream & istr, Bool clearheapp = False)
scan a frame from a stream

FrameSexp*

scan (Buffer * b, Bool clearheapp = False)
scan a frame from a buffer (string)

FrameSexp*
scan (Buffer & b, Bool clearheapp = False)
scan a frame from a buffer (string)

FrameSexp*
scan (const char * str, Bool clearheapp = False)
scan a frame from a c-string

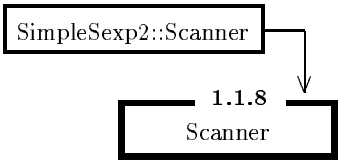
scan() facilities scans a frame from istrstream, Buffer, or c-string. These scan() uses heap's own scanner. This means that users need to take care of using the same heap under multi-threading.

1.1.8

class **Scanner** : public SimpleSexp2::Scanner

A Scanner of FrameSexp

Inheritance



Public Members

FrameSexp*
scan (Bool resetp = True)
scan a frame

A Scanner of FrameSexp

1.1.9

Usage

Names

1.1.9.1 **to scan a frame from a string.** 10
1.1.9.2 **to scan a frame from an istream** 11
1.1.9.3 **to scan a frame from a stream repeatedly** 11
1.1.9.4 **to scan a frame from a string.** 11
1.1.9.5 **to get a value of a slot** 12
1.1.9.6 **to create new atoms** 12
1.1.9.7 **to construct new frame** 13
1.1.9.8 **to set a slot to the frame** 13
1.1.9.9 **duplicate a new frame using a heap** 13

1.1.9.1

to scan a frame from a string.

to scan a frame from a string.

```
FrameSexp::Heap heap ;
FrameSexp* frame =
   heap.scan("(foo :bar (foo1 :bar () :baz 7) :boo 3
                   :a (:list a b c d e)),True) ;
if(!frame->chkSymtax()) { error(...) ; }
```

New cells and strings are allocated in `heap`. If the second argument of scan() is True, then the heap is cleared before scanning. If the second argument if False, the new cells and strings are allocated the rest area of the heap. All scan() does not check the syntax.

1.1.9.2

to scan a frame from an istream

to scan a frame from an istream

```
istream istr ;
...
FrameSexp::Heap heap ;
FrameSexp* frame =
   heap.scan(istr,True) ;
if(!frame->chkSymtax()) { error(...) ; }
```

1.1.9.3

to scan a frame from a stream repeatedly

to scan a frame from a stream repeatedly

```
istream istr ;
...
FrameSexp::Heap heap ;
FrameSexp::Scanner scanner(istr,False,&heap) ;
FrameSexp * data ;
for(data = scanner.scan() ; !data->isEof() ; data = scanner.scan()) {
   ...
}
```

this does not check the syntax.

1.1.9.4

to scan a frame from a string.

to scan a frame from a string.

```
FrameSexp* frame =
   FrameSexp::scan"(foo :bar (foo1 :bar () :baz 7) :boo 3
                    :a (:list a b c d e))") ;
if(!frame->chkSymtax()) { error(...) ; }
```

New cells and strings are allocated in SimpleSexp2::sharedheap (SimpleSexp2 is a parent class of FrameSexp.)

1.1.9.5

to get a value of a slot

to get a value of a slot

```
FrameSexp * value = frame->slotValue(":bar") ;

FrameSexp::Heap heap ;
FrameSexp * symbol = heap.newSymbol(":bar") ;
FrameSexp * value2 = frame->slotValue(symbol) ;
```

1.1.9.6

to create new atoms

to create new atoms

```
FrameSexp::Heap heap ;
FrameSexp * intval = heap.newInt(3) ;
FrameSexp * fltval = heap.newFlt(3) ;
FrameSexp * symval = heap.newSymbol("baz") ;
```

1.1.9.7

to construct new frame

to construct new frame

```
FrameSexp::Heap heap ;
FrameSexp frame = heap.newFrame("foo") ;
```

1.1.9.8

to set a slot to the frame

to set a slot to the frame

```
FrameSexp * slotname = heap.newSymbol(":bar") ;
FrameSexp * slotvalue = heap.newInt(3) ;
frame->setSlot(slotname, slotvalue,&heap) ;
```

setSlotValue() replaces the value of the slot. If the slot is not exists, the method add a new slot to the frame.

1.1.9.9

duplicate a new frame using a heap

duplicate a new frame using a heap

```
FrameSexp::heap1 ;
FrameSexp * original = heap1.scan("(foo :bar baz)") ;

FrameSexp::heap2 ;
FrameSexp * copied = heap2.dup(original) ;
```

While all data of the original frame is allocated in heap1, all data of copied frame is allocated in heap2. This is useful when a program use temporal and permanent data. In this case, users can prepare heaps for temporal and permanent use, and copy data useing dup() facilities between these heaps.

Class Graph

1.1 FrameSexp 2

1.1.7 Heap 7

1.1.8 Scanner 9