Introduction

The following tables provide a correlation between the concrete syntax from the SAL language report and the abstract synt
Document Type Definition (DTD). In some sense, the concrete syntax is simply the prettyprinted version of the DTD.

Here are some general principles:

- Attributes are used to keep information that is purely syntactical
- Restrict use of optionals to the ends of entities, e.g., in *ConstantDeclarations* there is an optional *VarDecls*, in the entity is not optional, but it may be empty.
- As much as possible, retain the names of the nonterminals in the concrete syntax
- The DTD is positional, generally the order of subentities in the DTD should reflect the order in the concrete syntax.

For example, there is no difference in semantics between the constant declarations

```
a, b: INTEGER;
c: INTEGER;
```

and

```
a: INTEGER;
b, c: INTEGER;
```

These are represented in the SAL DTD as:

```
<CONSTANTDECLARATION CHAIN="YES">
   <IDENTIFIER>a</IDENTIFIER>
   <VARDECLS></VARDECLS>
   <TYPENAME><SIMPLENAME>INTEGER</SIMPLENAME></TYPENAME>
</CONSTANTDECLARATION>
<CONSTANTDECLARATION CHAIN="NO">
   <IDENTIFIER>b</IDENTIFIER>
   <VARDECLS></VARDECLS>
   <TYPENAME><SIMPLENAME>INTEGER</SIMPLENAME></TYPENAME>
</CONSTANTDECLARATION>
<CONSTANTDECLARATION CHAIN="NO">
   <IDENTIFIER>c</IDENTIFIER>
   <VARDECLS></VARDECLS>
   <TYPENAME><SIMPLENAME>INTEGER</SIMPLENAME></TYPENAME>
</CONSTANTDECLARATION>
```

The second example would be the same, except that the first CHAIN would be NO, and the second would be YES.

Another example is infix applications, which don't directly exist in the DTD. Instead, there is an INFIX attribute that infix operation. Thus `A AND B` is treated as `AND(A, B)` with an INFIX attribute and translates to

```
<APPLICATION INFIX="YES">
  <NAMEEXPR><SIMPLENAME>AND</SIMPLENAME></NAMEEXPR>
  <TUPLELITERAL>
    <NAMEEXPR><SIMPLENAME>A</SIMPLENAME></NAMEEXPR>
    <NAMEEXPR><SIMPLENAME>B</SIMPLENAME></NAMEEXPR>
  </TUPLELITERAL>
</APPLICATION>
```

Contexts

Nonterminal	**SAL**	**XML**
Context	*Identifier* [{*Parameters*}] : CONTEXT = *ContextBody*	<CONTEXT> *Identifier Parameters ContextBo*
Parameters	[*TypeDecls*] TYPE ; $\{$*VarDecls*$\}^{*}_{,}$	<PARAMETERS>*TypeDecls VarDecls*</PARAMET
ContextBody	BEGIN *Declarations* END	<CONTEXTBODY> $\{$*Declarations*$\}^{+}$ </CONTEXT
Declarations	$\{$*Declaration* ;$\}^{+}$	$\{$*Declaration*$\}^{+}$
Declaration	*TypeDeclaration* | *ConstantDeclaration* | *AssertionDeclaration* | *ModuleDeclaration* | *ContextDeclaration*	*TypeDeclaration* | *ConstantDeclaration* | *AssertionDeclaration* | *ModuleDeclaration* | *ContextDeclaration*
TypeDeclaration	*Identifier* : TYPE [= *TypeDef*]	<TYPEDECLARATION> *Identifier* [*TypeDef*] </
TypeDef	*ScalarType* | *DataType* | *Type*	*ScalarType* | *DataType* | *Type*
ScalarType	{$\{$*Identifier*$\}^{+}_{,}$ }	<SCALARTYPE> $\{$*Identifier*$\}^{+}$ </SCALARTYPE>
DataType	DATATYPE *Constructors* END	<DATATYPE> *Constructors* </DATATYPE>
Constructors	$\{$*Constructor*$\}^{+}_{,}$	$\{$*Constructor*$\}^{+}$
Constructor	*Identifier* [(*Accessors*)]	<CONSTRUCTOR> *Identifier Accessors* </CONS
Accessors	$\{$*Accessor*$\}^{+}_{,}$	$\{$*Accessor*$\}^{+}$
Accessor	*Identifier* : *Type*	<ACCESSOR> *Identifier Type* </ACCESSOR>
ConstantDeclaration	*Identifier* [(*VarDecls*)] : *Type* [= *Expression*]	<CONSTANTDECLARATION> *Identifier Type* $\{$*Expression*$\}^{?}$ </CONSTANTDECLARATION>
AssertionDeclaration	*Identifier* : *AssertionForm* = *AssertionExpression*	<ASSERTIONDECLARATION> *Identifier Asserti* *AssertionExpression* </ASSERTIONDECLARA
ModuleDeclaration	*Identifier* [[*VarDecls*]] : MODULE = *Module*	<MODULEDECLARATION> *Identifier* $\{$*VarDecl*$\}^{*}$ *Module* </MODULEDECLARATION>
ContextDeclaration	*Identifier* : CONTEXT = *ContextName*	<CONTEXTDECLARATION> *Identifier ContextN*
ContextName	*Identifier* {*ActualParameters* }	<CONTEXTNAME> *Identifier ActualParameters*

Types

Nonterminal	SAL	XML
Type	*BasicType* | *Name* | *Subrange* | *ArrayType* | *TupleType* | *FunctionType* | *RecordType*	*BasicType* | *Name* | *Subrange* | *ArrayType* | *TupleTy* | *FunctionType* | *RecordType*
BasicType	`BOOLEAN` | `REAL` | `INTEGER` | `NZINTEGER` | `NATURAL` | `NZREAL`	`<BOOLEAN />` | `<REAL />` | `<INTEGER />` | `<NZINTEGER />` | `<NATURAL />` | `<NZREAL />`
Subrange	`[` *Bound* `..` *Bound* `]`	`<SUBRANGE>` *Bound Bound* `</SUBRANGE>`
Bound	*Unbounded* | *Expression*	*Unbounded* | *Expression*
Unbounded	`_`	`<UNBOUNDED />`
ArrayType	`ARRAY` *IndexType* `OF` *Type*	`<ARRAYTYPE>` *IndexType ElementType* `</ARRAYTY`
IndexType	`INTEGER` | *Subrange* | *ScalarTypeName*	`<INTEGER />` | *Subrange* | *ScalarTypeName*
ScalarTypeName	*Name*	*Name*
TupleType	`[` $\{$ *Type* $\}^{+}_{,}$ `]`	`<TUPLETYPE>` $\{$ *Type* $\}^{+}$ `</TUPLETYPE>`
FunctionType	`[` *Type* `->` *Type* `]`	`<FUNCTIONTYPE>` *Type Type* `</FUNCTIONTYPE>`
RecordType	`[#` $\{$ *FieldDecl* $\}^{+}_{,}$ `#]`	`<RECORDTYPE>` $\{$ *FieldDecl* $\}^{+}$ `</RECORDTYPE>`
FieldDecl	*Identifier* `:` *Type*	`<FIELDDECL>` *Identifier Type* `</FIELDDECL>`

Expressions

Nonterminal	SAL	XML
Expression	*NameExpr* | *NextVariable* | *Numeral* | *Application* | *InfixApplication* | *ArraySelection* | *RecordSelection* | *TupleSelection* | *UpdateExpression* | *LambdaAbstraction* | *QuantifiedExpression* | *LetExpression* | *SetExpression* | *ArrayLiteral* | *RecordLiteral* | *TupleLiteral* | *Conditional* | *ParenExpression*	*NameExpr* | *NextVariable* | *Numeral* | *Applic* | *InfixApplication* | *ArraySelection* | *Record* | *TupleSelection* | *UpdateExpression* | *Lam* | *QuantifiedExpression* | *LetExpression* | *S* | *ArrayLiteral* | *RecordLiteral* | *TupleLitera* | *Conditional* | *ParenExpression*
NextVariable	*Identifier* '	<NEXTOPERATOR> <NAMEEXPR>*Identifier*</N
Application	*Expression Argument*	<APPLICATION> *Expression Argument* </A
Argument	($\{Expression\}^{+}_{,}$)	<TUPLELITERAL> $\{Expression\}^{*}$ </TUPLEL
InfixApplication	*Expression Identifier Expression*	<APPLICATION INFIX="YES"> <NAMEEXPR> *Identifier* </NAMEEXPR> <TUPLELITERAL> *Expression Expression* </APPLICATION>
ArraySelection	*Expression*[*Expression*]	<ARRAYSELECTION> *Expression Expression* </ARRAYSELECTION>
RecordSelection	*Expression*.*Identifier*	<RECORDSELECTION> *Expression Identifier* </RECORDSELECTION>
TupleSelection	*Expression*.*Numeral*	<TUPLESELECTION> *Expression Numeral* </TUPLESELECTION>
UpdateExpression	*Expression* WITH *UpdatePosition* := *Expression*	<UPDATEEXPRESSION> *Expression UpdateS* </UPDATEEXPRESSION>
UpdatePosition	$\{Argument \mid [Expression] \mid .Identifier \mid .Numeral\}^{+}$	*See Below*
UpdateSelection	*See Below*	*Application* | *ArraySelection* | *RecordSelecti*

UpdateExpressions are represented as selections in the DTD. Thus

```
F WITH (x,y).a := 3
```

becomes

```
<UPDATEEXPRESSION>
  <NAMEEXPR>F<NAMEEXPR>
  <RECORDSELECTION>
    <APPLICATION>
      <NAMEEXPR>F</NAMEEXPR>
      <TUPLELITERAL>
        <NAMEEXPR>x</NAMEEXPR>
        <NAMEEXPR>y</NAMEEXPR>
```

```
      </TUPLELITERAL>
    </APPLICATION>
    a
  </RECORDSELECTION>
</UPDATEEXPRESSION>
```

Expressions (continued)

Nonterminal	SAL	XML
LambdaAbstraction	`LAMBDA (`*VarDecls*`) :` *Expression*	`<LAMBDAABSTRACTION>` *VarDecls Expression* `</LAMBDAABSTRACTION>`
QuantifiedExpression	*Quantifier* `(`*VarDecls*`) :` *Expression*	`<QUANTIFIEDEXPRESSION>` *VarDecls Expression* `</QUANTIFIEDEXPRESSION>`
LetExpression	`LET` *LetDeclarations* `IN` *Expression*	`<LETEXPRESSION>` *LetDeclarations Expression* `</LETEXPRESSION>`
LetDeclarations	$\{$*LetDeclaration*$\}^{+}_{,}$	$\{$*LetDeclaration*$\}^{+}$
LetDeclaration	*Identifier* `:` *Type* `=` *Expression*	`<LETDECLARATION>` *Identifier Type Expression* `</LETDECLARATION>`
SetExpression	*SetListExpression* | *SetPredExpression*	*SetListExpression* | *SetPredExpression*
SetPredExpression	`{`*Identifier* `:` *Type* `|` *Expression* `}`	`<SETPREDEXPRESSION>` *Identifier Type Expression* `</SETPREDEXPRESSION>`
SetListExpression	`{`$\{$*Expression*$\}^{+}_{,}$ `}`	`<SETLISTEXPRESSION>` $\{$*Expression*$\}^{+}$ `</SETLISTEXPRESSION>`
ArrayLiteral	`[ [` *IndexVarDecl* `]` *Expression* `]`	`<ARRAYLITERAL>` *IndexVarDecl Expression* `</ARRAYLITERAL>`
RecordLiteral	`(#` $\{$*RecordEntry*$\}^{+}_{,}$ `#)`	`<RECORDLITERAL>` $\{$*RecordEntry*$\}^{+}$ `</RECORDLITERAL>`
RecordEntry	*Identifier* `:=` *Expression*	`<RECORDENTRY>` *Identifier Expression* `</RECORDENTRY>`
TupleLiteral	`(` $\{$*Expression*$\}^{+}_{,}$ `)`	`<TUPLELITERAL>` $\{$*Expression*$\}^{+}$ `</TUPLELITERAL>`

Conditional	`IF` *Expression* `THEN` *Expression*
	$\{$`ELSIF` *Expression* `THEN` *Expression*$\}^{*}$
	`ELSE` *Expression* `ENDIF`
		`<CONDITIONAL>` *Expression Expression Expression* `</CO`
		`<CONDITIONAL ELSIF="YES">` *Expression Expression E*
		`</CONDITIONAL>`

ParenExpression	`(` *Expression* `)`	*See Below*

ParenExpression Every *Expression* element has a `PARENS` attribute that reflects the number of parentheses. Thus `((x +`

```
<APPLICATION INFIX="YES" PARENS="2">
  <NAMEEXPR>+</NAMEEXPR>
```

```
  <TUPLELITERAL>
    <NAMEEXPR>x</NAMEEXPR>
    <NUMERAL>1</NUMERAL>
  </TUPLELITERAL>
</APPLICATION>
```

Modules

Nonterminal	SAL	XML
Module	*BaseModule* | *SynchronousComposition* | *AsynchronousComposition* | *MultiSynchronous* | *MultiAsynchronous* | *Hiding* | *NewOutput* | *Renaming* | *ModuleName* | *ParenModule*	*BaseModule* | *SynchronousComposition* | *AsynchronousComposition* | *MultiSynchron* | *MultiAsynchronous* | *Hiding* | *NewOutput* | *Renaming* | *ModuleName* | *ParenModule*
BaseModule	`BEGIN` *BaseDeclarations* `END`	`<BASEMODULE>` *BaseDeclarations* `</BASEMODU`
BaseDeclarations	$\{$*BaseDeclaration*$\}^*$	$\{$*BaseDeclaration*$\}^*$
BaseDeclaration	*InputDecl* | *OutputDecl* | *GlobalDecl* | *LocalDecl* | *DefDecl* | *InitDecl* | *TransDecl*	*InputDecl* | *OutputDecl* | *GlobalDecl* | *LocalDecl* | *DefDecl* | *InitDecl* | *TransDecl*
InputDecl	`INPUT` *VarDecls*	`<INPUTDECL>` *VarDecls* `</INPUTDECL>`
OutputDecl	`OUTPUT` *VarDecls*	`<OUTPUTDECL>` *VarDecls* `</OUTPUTDECL>`
GlobalDecl	`GLOBAL` *VarDecls*	`<GLOBALDECL>` *VarDecls* `</GLOBALDECL>`
LocalDecl	`LOCAL` *VarDecls*	`<LOCALDECL>` *VarDecls* `</LOCALDECL>`
DefDecl	`DEFINITION` *Definitions*	`<DEFDECL>` *Definitions* `</DEFDECL>`
InitDecl	`INITIALIZATION` $\{$*DefinitionOrCommand*$\}^+_;$	`<INITDECL>` $\{$*DefinitionOrCommand*$\}^+$ `</IN`
TransDecl	`TRANSITION` $\{$*DefinitionOrCommand*$\}^+_;$	`<TRANSDECL>` $\{$*DefinitionOrCommand*$\}^+$ `</T`
DefinitionOrCommand	*Definition* | `[` *SomeCommands* `]`	*Definition* | *SomeCommands*
Definition	*SimpleDefinition* | *ForallDefinition*	*SimpleDefinition* | *ForallDefinition*
SimpleDefinition	*Lhs RhsDefinition*	`<SIMPLEDEFINITION>` *Lhs RhsDefinition* `</SI`
Lhs	*Identifier* [`'`] $\{$*Access*$\}^*$	*NextVariable* | *ArraySelection* | *RecordSelection*
Access	*ArrayAccess* | *RecordAccess* | *TupleAccess*	
ArrayAccess	`[` *Expression* `]`	
RecordAccess	`.` *Identifier*	
TupleAccess	`.` *Numeral*	
RhsDefinition	*RhsExpression* | *RhsSelection*	*RhsExpression* | *RhsSelection*
RhsExpression	`=` *Expression*	`<RHSEXPRESSION>` *Expression* `</RHSEXPRESSI`
RhsSelection	`IN` *Expression*	`<RHSSELECTION>` *Expression* `</RHSSELECTION`
ForallDefinition	`(` `FORALL (` *VarDecls* `) :` *Definitions* `)`	`<FORALLDEFINITION>` *VarDecls Definitions* `</`
Definitions	$\{$*Definition*$\}^+_;$	$\{$*Definition*$\}^+$
SomeCommands	$\{$*SomeCommand*$\}^+_{[]}$	$\{$*SomeCommand*$\}^+$

Lhs An *Lhs* of the form `x'[3].a` corresponds to

```
<RECORDSELECTION>
  <ARRAYSELECTION>
    <NEXTOPERATOR><NAMEEXPR>x</NAMEEXPR></NEXTOPERATOR>
```

```
    <NUMERAL>3</NUMERAL>
  </ARRAYSELECTION>
  a
</RECORDSELECTION>
```

Modules (continued)

Nonterminal	SAL	XML
SomeCommand	*NamedCommand* | *MultiCommand*	*NamedCommand* | *MultiCommand*
NamedCommand	[*Identifier* :] *GuardedCommand*	<GUARDEDCOMMAND LABEL="Identifier"> </GUARDEDCOMMAND>
GuardedCommand	*Guard* --> *Assignments*	*Guard Assignments*
Guard	*Expression*	<GUARD> *Expression* </GUARD>
Assignments	$\{$*SimpleDefinition*$\}^{+}_{;}$	<ASSIGNMENTS> $\{$*SimpleDefinition*$\}^{+}$ </A
MultiCommand	([] (*VarDecls*) : *SomeCommand*)	<MULTICOMMAND> *VarDecls SomeComman*
SynchronousComposition	*Module* || *Module*	<SYNCHRONOUSCOMPOSITION> *Module Mo* </SYNCHRONOUSCOMPOSITION>
AsynchronousComposition	*Module* [] *Module*	<ASYNCHRONOUSCOMPOSITION> *Module M* </ASYNCHRONOUSCOMPOSITION>
MultiSynchronous	(|| (*Identifier* : *Subrange*) : *Module*)	<MULTISYNCHRONOUS> *Identifier Subrange* </MULTISYNCHRONOUS>
MultiAsynchronous	([] (*Identifier* : *Subrange*) : *Module*)	<MULTIASYNCHRONOUS> *Identifier Subrang* </MULTIASYNCHRONOUS>
Hiding	LOCAL $\{$*Identifier*$\}^{+}_{,}$ IN *Module*	<HIDING> $\{$*Identifier*$\}^{+}$ *Module* </HIDING
NewOutput	OUTPUT *VarDecls* IN *Module*	<NEWOUTPUT> *VarDecls Module* </NEWOUT
Renaming	[WITH *NewVarDecls*] RENAME *Renames* IN *Module*	<RENAMING> *NewVarDecls Renames Modu*
NewVarDecls	$\{$*InputDecl* | *OutputDecl* | *GlobalDecl*$\}^{+}_{;}$	<NEWVARDECLS> $\{$*InputDecl* | *OutputDecl* | </NEWVARDECLS>
Renames	$\{$*Lhs* TO *Lhs*$\}^{+}_{,}$	<RENAMES> *Lhs Lhs* </RENAMES>
ModuleName	*Name* [[$\{$*Expression*$\}^{+}_{,}$]]	<MODULENAME> *Name* $\{$*Expression*$\}^{+}$ </MO
ParenModule	(*Module*)	<PARENMODULE> *Module* </PARENMODULE>

Misc

Nonterminal	SAL	XML
TypeDecls	$\{$*Identifier*$\}_{,}^{+}$ `: TYPE`	`<TYPEDECLS>` $\{$*Identifier*$\}^{*}$ `</TYPEDECLS>`
VarDecls	$\{$*VarDecl*$\}_{,}^{+}$	`<VARDECLS>` *VarDecl* `</VARDECLS>`
VarDecl	$\{$*Identifier*$\}_{,}^{+}$ `: Type`	`<VARDECL CHAIN="NO">` *Identifier Type* `</VARDECL>`
Name	*SimpleName* | *QualifiedName*	*SimpleName* | *QualifiedName*
SimpleName	*Identifier*	`<SIMPLENAME>` *Identifier* `</SIMPLENAME>`
QualifiedName	*ContextName Identifier*	`<QUALIFIEDNAME>` *ContextName Identifier* `</QUALIFIEDN`
ContextName	*Identifier* [`{` *ActualParameters* `}`]	*¡CONTEXTNAME¿ Identifier ActualParameters ¡/CON*
Identifier	{ *Letter* {*Letter* | *Digit* | `?` | `_`}* } | {*Opchar*}$^{+}$	`(#PCDATA)`