Behavior Tree Notation v1.0 (2007)

Behavior Tree Group
ARC Center for Complex Systems

March 15, 2007

Contents

Chapter 1

Naming Conventions

1.1 Variable Naming Conventions

Variable	**Description**
N, N_i	Behavior Tree Nodes
T, T_i	Behavior Trees
C, C_i	Components
$C\#$	A Component Instance
s	A State of a Component
e	An Event
a	An Attribute of a Component
b	A Branching Condition of a Component

Table 1.1: Variable Naming Conventions

1.2 Node Concrete Syntax

Label	Name	Description
A	Component Name	Specifies a component
B	Behavior	Specifies the behaviour associated with the component
C	Operator(s)	Indicates behaviour of this node is dependent on another node in the tree
D	Label	An optional label for disambiguation (in case a node appears elsewhere with the same component and behaviour)
E	Behavior Type	Delimiters on the behaviour indicate the type of behaviour involved
F	Traceability Link	A reference to the requirements document
G	Traceability Status	Indicates how the node relates to the link
H	Tag	The box on the left-hand side of the node (by default, contains traceability information, but may be used differently, or omitted, in different contexts)
I	Behavior Tree Node	

Table 1.2: Elements of a Behavior Tree Node

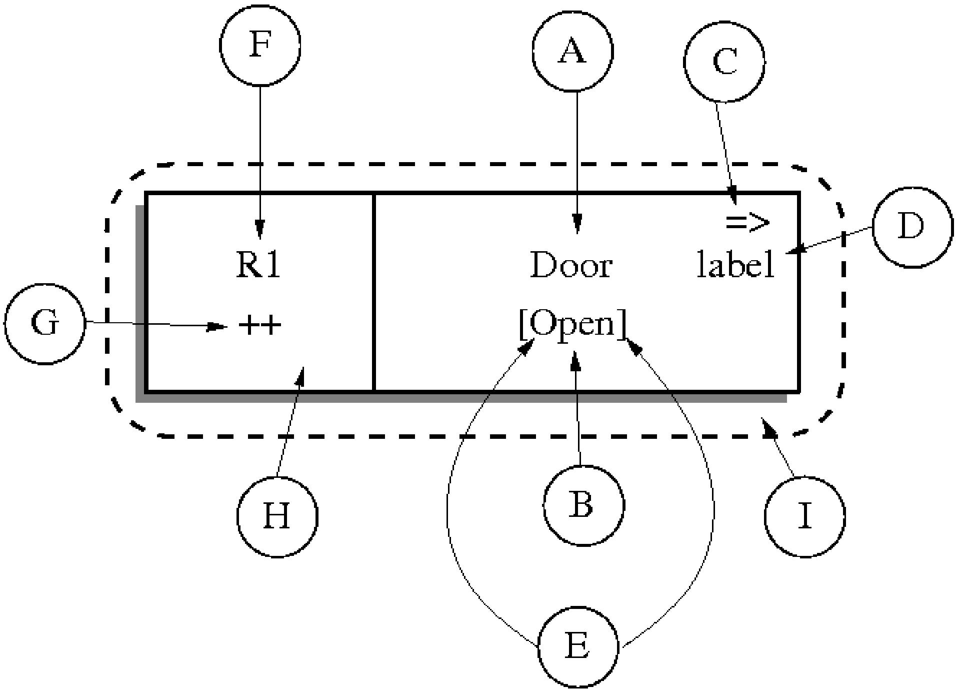


Figure 1.1: BT Node concrete syntax example

1.3 Tree Naming Conventions

The following conventions are used to refer to nodes relative to a node of interest.

Label	Name	Description
A	Ancestor Node	Any node which appears in a direct line between the node of interest and the root node of the tree
B	Parent Node	An immediate ancestor
C	Sibling Node	A node which shares the same parent
D	Sibling Branch	A (sub)tree with a sibling node as its root
E	Child Node	A node immediately below
F	Descendant	Any node appearing below

Table 1.3: Nodes of a Behavior Tree

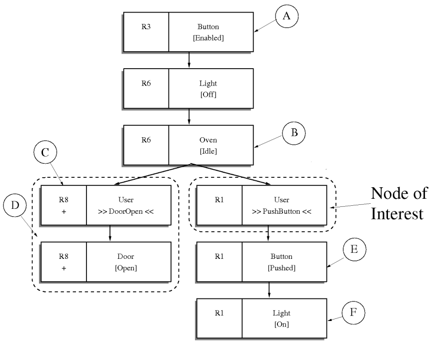


Figure 1.2: Behavior Tree 'Tree' Naming Conventions

1.4 Tree Branch Naming Convention

The following conventions are used to refer to branches of a tree relative to a node of interest.

Label	Name	Description
A	Root Node	The first node in a tree (does not have a parent)
B	Edge	
C	Leaf Node	A node with node children
D	Subtree	A tree contained within the tree rooted at the node of interest
	Branch	A synonym for subtree

Table 1.4: Branches of a Behavior Tree

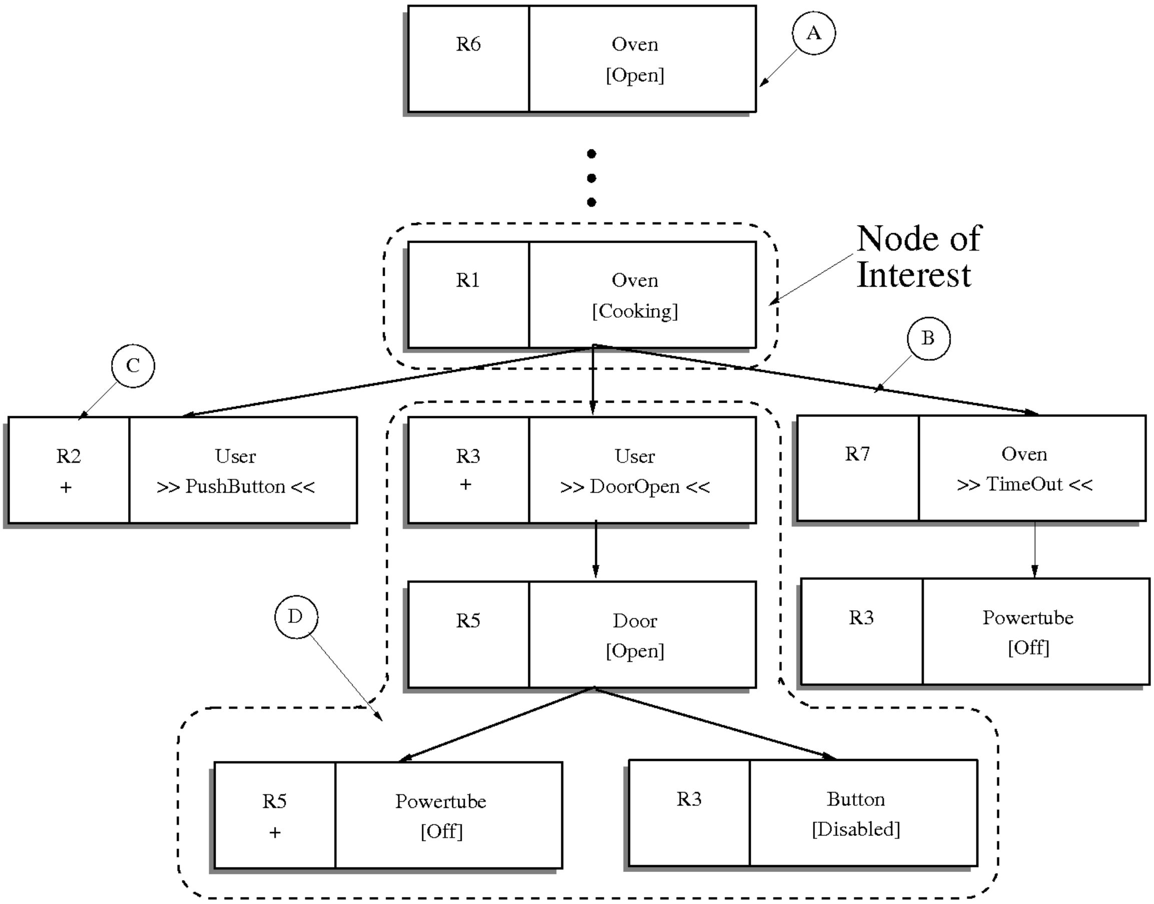
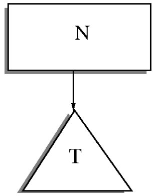
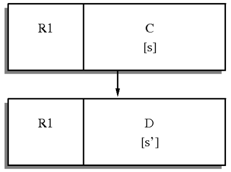
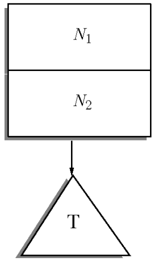
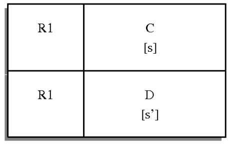
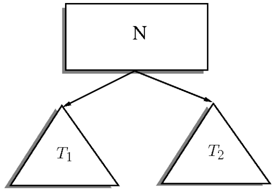
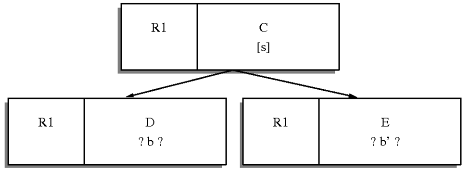
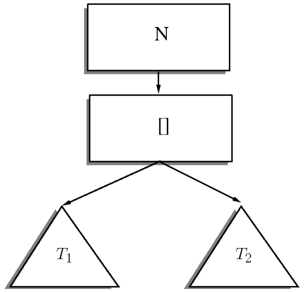
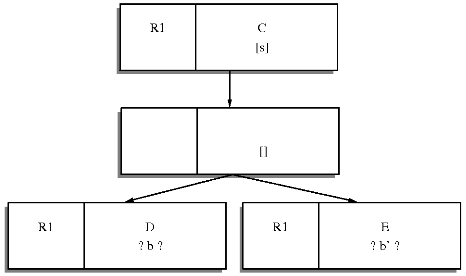


Figure 1.3: Tree Branch Naming Convention

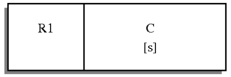
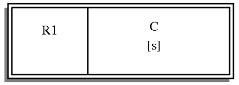
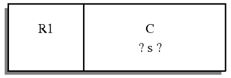
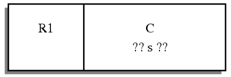
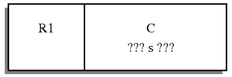
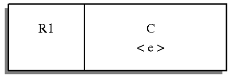
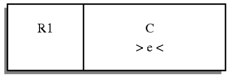
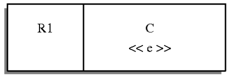
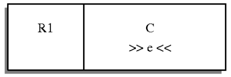
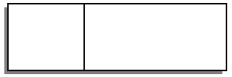
Chapter 2

Behavior Tree Notation

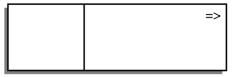
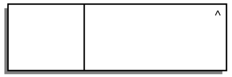
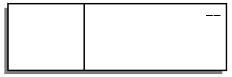
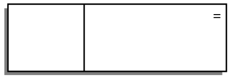
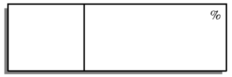
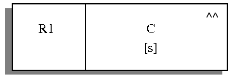
2.1 Behavior Tree Composition

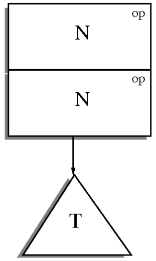
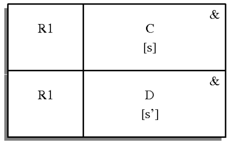
Type	General Definition	Example	Textual Notation	Description
Sequential Composition			$N; T$	Execute N, passing control to tree T. The behaviour of concurrent BTs may be interleaved between N and T.
Atomic Composition			$N_1;;(N_2;T)$	Execute N_1 immediately followed by N_2, passing control to tree T. The behaviour of concurrent BTs may not be interleaved between N_1 and N_2.
Parallel Branching			$N; (T_1 \| \| T_2)$	Execute N, passing control to both T_1 and T_2.
Alternative Branching		 R1 C [s] [] R1 D ? b ? R1 E ? b' ?	$N; (T_1 [] T_2)$	A nondeterministic choice is made between T_1 and T_2, depending on which is ready to execute (not blocked)

2.2 Basic nodes

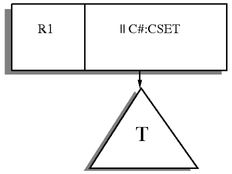
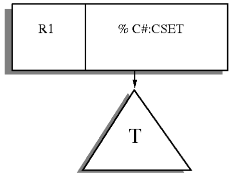
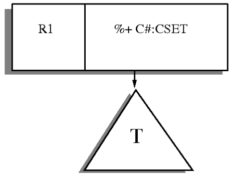
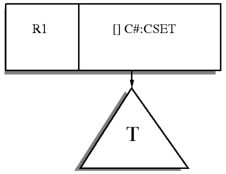
Type	Graphical Notation	Textual Notation	Description
State Realisation		C[s]	Component C realises state s
System State Realisation		C[s]	This is a state realisation decorated with a double box to indicate the component is system component in the current context. There can only be one system component in each context.
Selection		$C?s?$	Special
Event		$C??e??$	Wait until event e is received
Guard		$C???s???$	Wait until C is in state S
Internal Output Event		send $C.e$	Generate event e internally to system
Internal Input Event		recv $C.e$	Wait for event e (from system)
External Output Event			Generate event e and send to environment
External Input Event			Wait for event e to be received from environment
Empty Node		skip	Empty Nodes when used with labels can be used as origins or destinations of node operators.

2.3 Node Operators

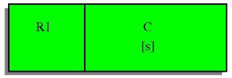
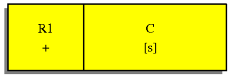
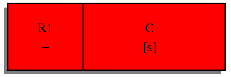
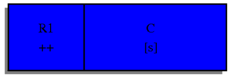
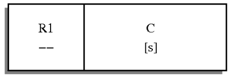
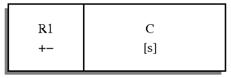
Type	Graphical Notation	Textual Notation	Description
Reference		$N^{=>}$	Behave as the destination tree. The destination node must appear in an alternative branch to the origin.
Reversion		$N^{\wedge}$	Behave as the destination tree. The destination node must be an ancestor. All sibling behaviour is terminated.
Branch Kill		N^{--}	Terminate all behaviour associated with destination tree
Synchronisation		$N^{=}$	Wait for destination node (or nodes)
May		$N^{\%}$	The node may execute normally, or may have no effect
Start new		$N^{\wedge\wedge}$	As with reversion, but sibling behaviour is not terminated

Condition operators			N_1 op N_2	The operator op may be one of &, |, or XOR, corresponding to logical conjunction, disjunction and exclusive or

2.4 Multiple Component Instances

Type	Graphical Notation	Description
For All		Execute an instance of T for every element in $CSET$
For Some		Execute an instance of T for some number (including 0) of elements in $CSET$
At Least One		Execute an instance of T for some number (but at least 1) of elements in $CSET$
For One Arbitrary		Execute an instance of T for one element in $CSET$

2.5 Node Tags

Type	Graphical Notation	Description
Original Behavior		No Traceability Status indicates that the behavior is stated in the original requirements. The color "green" is used for original requirements.
Implied Behavior		The "+" Traceability Status indicates that the behavior is not explicitly stated in the original requirement but is implied by the requirement. The color "yellow" is used for implied behavior.
Missing Behavior		The "-" Traceability Status indicates that the behavior is missing from the original requirements and is needed for completeness. The color "red" is used for missing behavior.
Updated Behavior		The "++" Traceability Status indicates that the behavior has been added in the post-development (PD) or maintainence phase. The color "blue" is used for updated behavior. Where there are different series of changes / upgrades we use ++V1.0, ++V2.0, etc to indicate the particular upgrade series.
Deleted Behavior		No Descriptions Yet
Design Refinement Behavior		The "+-" Traceability Status indicates that the behavior is a refinement of the original requirements, indicating that the behavior is implied but the detail to describe it is missing.

Appendix A

Expression syntax

The expressions that may appear in the *Behavior* of a node are drawn from a rich logical language. The proviso is that the operators must appear on a keyboard, i.e., do not make use of general mathematical notation. Overloading may occur, for example, the operation ‘+’ may refer to integer addition or to set union, depending on the context.

A.1 Expression grammar

The grammar is given below in EBNF syntax (suitable for the GOLD parser).

```
Num = {Number}+
Name = {Letter}{AlphaNumeric}*
Not = NOT

! ------------------------------------------------- Terminals
OpenCurly = '{'
CloseCurly = '}'
Plus = '+'
Minus = '-'
Intersect = '><'
LessThan = '<'
GreaterThan = '>'
EqualTo = '='
Colon = ':'
Pipe = '|'
OpenSquare = '['
CloseSquare =']'
OpenRound = '('
CloseRound = ')'
Comma = ','
LessThanOrEqual='=<'
GreaterThanOrEqual='>='
AssignExp = ':='
SubsetExp = '<:'
EventDelim = '??'
! ------------------------------------------------- Rules

<BhvExp> ::=    <StateRealisation> | <AttributeRealisation> | <Condition>
    | <Event>
    | <forAll> | <forSome>

<StateRealisation> ::= <Exp>
<AttributeRealisation> ::= <ident> AssignExp <Exp>
```

```
<Exp> ::=  <ident> | <Number> | <fnapp> | <bracketedexp> | <binaryexp>
          | <card> | <enumeratedset> | <empty>  | <FuzzyExp>

<FuzzyExp> ::= CloseSquare <Exp> OpenSquare

<binaryexp> ::= <Exp> <binaryop> <Exp>

<binaryop> ::= Plus | Minus | Intersect

<card> ::= Pipe <Exp> Pipe

<enumeratedset> ::= OpenCurly <Explist> CloseCurly

<empty> ::= OpenCurly CloseCurly

<fnapp> ::= <ident> OpenRound <Explist> CloseRound
         | <ident> OpenRound CloseRound

<Condition> ::= <binaryreln> | <notcondition> | <bracketedcondition>

<binaryreln> ::= <Exp> <binarysym> <Exp>

<binarysym> ::= LessThan | GreaterThan | EqualTo | LessThanOrEqual
         | GreaterThanOrEqual | Colon | SubsetExp

<notcondition> ::= Not <Condition>

<bracketedcondition> ::= OpenRound <Condition> CloseRound

<bracketedexp> ::= OpenRound <Exp> CloseRound

<Event> ::= EventDelim <fnapp> EventDelim

<forAll> ::=  Pipe Pipe <ident> Colon <Exp>

<forSome> ::= OpenSquare CloseSquare <ident> Colon <Exp>

<ident> ::= Name | Name . Name | Name . Name . Name

<Number> ::= Num

<Explist>
        ::= <Exp>
         |  <Exp> Comma <Explist>
```

A.2 Interpretations

The operators of the language are interpreted differently depending on the type of the components in question (as determined by the composition tree).

A.2.1 Numbers

The mathematical operators are given their usual interpretation.

A.2.2 Sets

Operator	Interpretation for sets
+	Set union
-	Set difference
><	Set intersection

Appendix B

Glossary

ASG : Abstract Syntax Graph

BT(s) : Behavior Tree(s)

BT Graphical Notation: A Visual representation of Behavior Trees.

BT Textual Notation: An equivalent of the Behavior Tree Graphical notation in textual form

BT Edge: Connection (line) between Behavior Tree nodes.

BT Node: Fundamental unit of a Behavior Tree.

Branching: Represented as multiple edges, Branching can be concurrent, modelling parallel behavior of threads, or alternative, where only one of the branches can succeed.

CSG : Concrete Syntax Graph

Destination : The node referred to by a reference node

Origin : A reference node (contains an operator specifying macro, kill, revert, synchronise, etc.)