Towards an Interchange Standard for Editable Documents

by Jim Mitchell and Jim Horning

August 31, 1982 4:19 PM

File: Interscript-app.bravo

APPENDIX A

GLOSSARY

Italics indicate words defined in this glossary.

abbreviation An *invocation* used to shorten a *script*, rather than to indicate structure

attribute A component of an *environment*, identified by its *name*, which is bound to a *value*

base language The part of the *Interscript* language that is independent of the semantics of particular *properties* and *attributes*

base semantics The semantic rules that govern how *scripts* in the *base language* are elaborated to determine their *contents, environments,* and *labels*

binding The operation of associating a *value* with a *name* to add an *attribute* to an *environment*; also the resulting association

binding mode A *value* may be bound to an *identifier* as *local, const* or *global*

Boolean An enumerated *primitive type* (F, T) used to control *selection* and as *primitive values*

const binding A *binding* of an *attribute* that prevents its being rebound in any contained *scope*

contents The *vector* of *values* denoted by a *node* of a *script*

definition Another name for a *const binding*

document The *internalization* of a *script* in a representation suitable for some editor

dominant structure The tree structure of a *document* corresponding to the *node* structure of its *script*

editor-specific name A non-standard *name* used by a specific editor in *scripts* it generates; an editor may use editor-specific terms without interfering with the interchangeability of a script if it provides *definitions* of the standard names in terms of its editor-specific names

elaborate (verb) To develop the semantics of a *script* or a *node* of a script according to the *Interscript* semantic rules. This is a left-to-right, depth-first processing of the script

encoding A particular representation of *scripts*

environment A *value* consisting of a set of *attributes*. An environment may be either *free-standing* or *nodal*. A free-standing environment is a structured value much like a record, with the components being the *attributes* of the environment. A nodal environment is associated with a *node* of a *script* and represents the *attributes* bound in that node.

expression A syntactic form denoting a *value*

external environment A standard *environment* relative to which an entire *script* is *elaborated*

externalization The process of converting from a *document* to a *script*; also the result of that process

fidelity The extent to which an *externalization* or *internalization* preserves *contents*, form, and structure

hexInt A component of a *hexSequence* formed from a pair of letters in the set {A,B,…,O,P}, and representing an *integer* in the range [0..256)

hexSequence A sequence of *hexInt* pairs enclosed between "#" pairs and used to encode characters in *string* literals, e.g., #ENCODE#

hierarchical name A *name* containing at least one period, whose prefix unambiguously denotes the naming authority that assigned its meaning

identifier A sequence of letters used to identify an *attribute*

integer A mathematical integer in a limited range; one of the *primitive types*

interchange encoding The standard *encoding* for *scripts*

internalization The process of converting from a *script* to a *document*; also the result of that process

Interscript The current name of this basis for an editable document standard

invocation The appearance of a *name* in an *expression*, except as the *attribute* of a *binding*

label A *tag,* or a *source*, a *target*, or a *link introduction* placed in a *node*

link The cross product of a *source* and a *target*; in general, a link is a set of (source, target) pairs; in the special case when there is exactly one source and one target, a link behaves like a directed arc between a pair of *nodes*

link introduction The appearance of LINKS id in a *node*, where id is the main *identifier* of a *link*

literal A representation of a *value* of a *primitive type* in a *script*

local binding A *binding* of a *value* to a *name*, causing the current *environment* to be updated with the new *attribute*; any outer binding's *scope* will resume at the end of the innermost containing *node*

name A sequence of *identifiers* internally separated by periods; e.g., a.b.c

nested environment The initial *environment* of a *node* contained in another node

NIL A name for the empty *value*; it does not lengthen a *vector* or *node* in which it appears

node Everything between a matched pair of {}s in a *script*; this generally represents a branch point in a *document's dominant structure*

NULL Identifies the empty *environment*; the *value* it associates with any *identifier* is *NIL*

OUTER A standard *attribute* of every *environment:*

For a *free-standing environment* (i.e., a record-like, structured value), OUTER=NULL

For a *nodal environment*, OUTER's value is the *environment* of the current node's parent just prior to the start of the current *node*.

For the root node of a document, OUTER=*X*.

For *X*, OUTER=NULL

global binding A kind of *binding* (indicated by ":=") that modifies the environment of the root node of a document only, and hence may endure beyond the end of the current node and may be seen by nodes to the right of the current node, even those not hierarchically descended from the current node.

primitive type *Boolean*, *Integer*, *Real*, *String*, or *Universal*

primitive value A *literal* or a *node*, *vector*, or *environment* containing only primitive values

private encoding One of a number of non-standard *encodings* of a *script*

property Each *tag* on a *node labels* it with a *property*; the *properties* of a *node* determine how it may be viewed and edited

quoted expression A *value* which is an *expression* bracketted by single quotes ("'"); the expression is evaluated in each *environment* in which the *identifier* to which it is *bound* is *invoked*

real A floating point number

scope The region of the *script* in which *invocations* of the *attribute* named in a *binding* yield its *value*; the *scope* starts textually at the end of the *binding*, and generally terminates at the end of the innermost containing *node*

script An *Interscript* program; the interchangeable result of *externalizing* a *document*

selection A conditional form in a *script* that denotes one of two *expressions*, depending on the value of a *Boolean* expression in the current *environment*

source The set of *nodes* with REF *link*, which thereby refer to the set of *target links.*

string A *literal* which is a *vector* of characters bracketed by "<>", e.g., <This is a string!>

style A *quoted expression* to be invoked in a *node* to modify the node's *environment*, *labels*, or *contents*

Sub A standard component of each *environment*, which is implicitly invoked to initialize *nested environments*

SUBSCRIPT	A function that can be used to extract a value from a *vector*, e.g. SUBSCRIPT[(a b <str>), 3] is the value <str>
tag	A *universal name labelling* a *node* using the syntax universal$; the *properties* of a node correspond to the set of tags labelling it
target	The set of *nodes labelled* with *link*:
transparency	A characteristic of *scripts* that allows an editor to identify the *nodes* of a script that it understands and thereby enables it to operate on those nodes without disturbing the ones that it doesn't understand
Units	A set of *definitions* relating various typographical and scientific units to the *Interscript* standard units, meters; e.g., inch=2.54E 2*meter, pt=.013836*inch
universal	An *identifier* formed entirely of uppercase letters and digits
value	A *primitive value*, *node*, *vector*, *environment*, *universal*, or *quoted expression*
vector	An ordered sequence of *values* that may be *subscripted*
X	The standard outer *environment* for an entire *script*; the *value* of an unbound *identifier* in X is the *universal* consisting of the same letters in upper case

APPENDIX B

ARBITRARY CHOICES

"One of the primary purposes of a standard
is to be definitive about otherwise arbitrary choices."

There are many places in this proposal where we have made an arbitrary choice for definiteness. It will be important that the ultimate standard make *some* choice on these points; it matters little whether it is the same as ours. To forestall profitless debate on these points, we have tried to list some of the choices that we believe can be easily changed at a later date:

Encoding choices:

The choice of representations for literals (we generally followed Interpress here).

The selection of particular characters for particular kinds of bracketting, and for particular operators.

The choice of infix and functional notation for the interchange encoding (as opposed, e.g., to Polish postfix).

The choice of particular identifiers for basic concepts.

Linguistic choices:

The choice of a particular set of basic operators for the language.

The particular set of primitive data types (we followed Interpress its set seems about as small as will suffice).

The choice of particular syntactic sugars for common linguistic forms.

APPENDIX C

FORMAL SEMANTICS

C.1. Grammar

Our notation is basically BNF with terminals quoted and augmented by the following conventions:

a sequence enclosed in [ ] brackets may occur zero or one times;

a construct followed by * may occur zero or more times;

parentheses ( ) are used purely for grouping.

script	::=	header node trailer
header	::=	"Interscript/Interchange/1.0 "
trailer	::=	"EndScript"
item	::=	content | binding | label
content	::=	term | node
term	::=	primary | primary op term
op	::=	"+" | " " | "*" | "/"
primary	::=	literal | invocation | indirection | application | selection | vector
literal	::=	Boolean | integer | real | string | universal
invocation	::=	name
name	::=	id ("." id)*
indirection	::=	name "%"
application	::=	(name | universal) "[" item* "]"
universal	::=	ucID
selection	::=	"(" term "|" item* "|" item* ")"
vector	::=	"(" item* ")"
node	::=	"{" item* "}"
binding	::=	localBind | globalBind
localBind	::=	name "_" rhs
globalBind	::=	(name | universal) ":=" rhs
rhs	::=	content | op term | "'" item* "'" | "[" item* "|" binding* "]"
label	::=	tag | link
tag	::=	universal "$"
link	::=	"LINKS" id | "^" name | name ":"

C.2. Notation for environments

Environments bind identifiers to expressions, in various modes ("=", ":=", "_"):

NULL denotes the "empty" environment

[*E* | *id* _ *e*] means "*E* with *id* bound to *e*"

locVal(*id*, *E*) denotes the value locally bound to *id* in *E*

locVal(*id*, NULL) = NIL = ""

locVal(*id*, [*E* | *id'* m *e*]) = if *id*=*id'* then *e* else *locVal*(*id*, *E*)

C.3. Semantic functions

R: expression, environment --> expression -- Reduction

R is used for evaluating right-hand sides: identifiers, expressions, etc.

C: expression --> expression -- Contents

C is basically used to indicate which evaluated expressions become part of the content of a node

B: expression, environment --> environment -- Bindings

B indicates the effect a binding has on an environment. B and R are mutually recursive functions (e.g., the evaluation of an expression may cause some bindings to occur as well)

The following four semantic functions occur less frequently in any substantive way in the semantics below. You might wish to skip them until they occur in a nontrivial manner in the semantics.

T: expression --> expression -- Tags

T indicates when an identifier is to be included in the tag set for a node

L: expression --> expression -- Links

L indicates link declarations

L_S: expression --> expression -- Link sources

L_S indicates a link to the set of nodes having associated target links

L_t: expression --> expression -- Link targets

L_t indicates that the node is to be included in the target set of all the names which are prefixes of the name to which the expression should evaluate

C.4. Presentation by feature

[*E* is used to represent the value of the environment in which the feature occurs.]

script ::= header node trailer
header ::= "Interscript/Interchange/1.0 "
trailer ::= "EndScript"

The semantics of the root node of a script are equivalent to the following general semantics for a node with the initial environment being the outermost, external environment X instead of *E*:

node ::= "{" *item** "}"
R = *C* = "{" *R*<"Sub" *item**>([NULL | "OUTER" "=" *E*]) "}"
B = *locVal*("OUTER", (*B*<"Sub" *item**>([NULL | "OUTER" "=" *E*])))
T = *L* = L_S = L_t = NIL

Nodes have nested environments, and can have more global effects only through global (:=)

bindings. The items of a node are implicitly prefixed with the identifier Sub, which may be bound to any information intended to be common to all subnodes in a scope.

*item** ::= ""
    $R = C = T = L = L_S = L_t$ = NIL
    $B = E$

The empty sequence of items has no value and no effect; this is the basis for the following recursive definition.

*item** ::= *item1 item**
    R = *R*<*item1*>(*E*) *R*<*item**>(*B*<*item1*>(*E*))
    B = *B*<*item**>(*B*<*item1*>(*E*))
  For *F* in {*C*, *T*, *L*, L_S, L_t}:
    F = *F*<*item1*> *F*<*item**>

In general, the value of a sequence of items is just the sequence of item values; binding items affect the environment of items to their right; NIL does not change the length of a result sequence.

term ::= *primary op term*
op ::= "+" | "-" | "*" | "/"
    R = *C* = *R*<*primary*>(*E*) *op R*<*term*>(*E*)
    B = *E*
    $T = L = L_S = L_t$ = NIL

Both the primary and the term must reduce to numbers; the arithmetic operators are evaluated right-to-left (a la APL, without precedence) and bind less tightly than application.

primary ::= *literal*
literal ::= *Boolean* | *integer* | *hexint* | *real* | *string*
    R = *C* = *literal*
    B = *E*
    $T = L = L_S = L_t$ = NIL

The basic contents of a document.

invocation ::= *id*
    R = *R*<*valOf*(*id*, *E*)>(*E*)
    B = *B*<*valOf*(*id*, *E*)>(*E*)
  where
    valOf(*id*, *E*) = *locVal*(*id*, *whereBound*(*id*, *E*)) -- Gets innermost value
    whereBound(*id*, *E*) = CASE -- Gets innermost binding
      locBinding(*id*, *E*) ~= NONE => *E*
      locBinding("OUTER", *E*) ~= NONE =>
        whereBound(*id*, *locVal*("OUTER", *E*))
      True => NULL

Both attributes and definitions are looked up in the current environment; depending on the current binding of *id*, this may produce values and/or bindings; if the binding's *rhs* was quoted, the expression is evaluated at the point of invocation.

When an *id* is referred to and *locBinding*(*id*, *E*)=NONE, then the value is sought recursively in *locVal*("OUTER"). The outermost environment, *X*, binds each *id* to the "universal" name which is the uppercase equivalent of *id*.

invocation ::= *name* "." *id*
R = *R*<*valOf*(*id*, *R*<name>(*E*))>(*E*)
B = *B*<*valOf*(*id*, *R*<name>(*E*))>(*E*)

Qualified names are treated as "nested" environments.

universal ::= *ucID*
R = *C* = *ucID*
B = *E*
T = *L* = L_S = L_t = NIL

Uppercase-only identifiers are presumed to be directly meaningful and are not looked up in the environment.

application ::= *invocation* "[" *item** "]"
R = *apply*(*invocation*, *R*<*item**>(*E*), *E*)
B = *E*
where
apply(*invocation*, *value**, *E*) =
CASE *R*<*invocation*>(*E*) OF
"EQUAL" => *value1* = *value2*
"GREATER" => *value1* > *value2*
. . .
"SUBSCRIPT" => *value1*[*value2*] -- value1: sequence, value2: int
"CONTENTS" => "(" C<*inner*(*value1*)> ")"
"TAGS" => "(" T<*inner*(*value1*)> ")"
"LINKS" => "(" L<*inner*(*value1*)> ")"
"SOURCES" => "(" L_S<*inner*(*value1*)> ")"
"TARGETS" => "(" L_t<*inner*(value1)> ")"
ELSE => *R*<*invocation*>([[NULL | "OUTER" "=" *E*] | "Value" "=" *value**])
inner("{" *value** "}") = *value**

If the *invocation* does not evaluate to one of the standard external function names, the current environment is augmented with a binding of the value of the argument list to the identifier *Value*, and the value is the result of the invocation in that environment; this allows function definition within the language.

selection ::= "(" *term* "|" *item1** "|" *item2** ")"
R = if *R*<*term*>(*E*) then *R*<*item1**>(*E*) else *R*<*item2**>(*E*)
B = if *R*<*term*>(*E*) then *B*<*item1**>(*E*) else *B*<*item2**>(*E*)

The notation for selections (conditionals) is borrowed from Algol 68:
(<*test*> | <*true part*> | <*false part*>)
This is consistent with our principles of using balanced brackets for compound constructions and avoiding syntactically reserved words; the *true part* and *false part* may each contain an arbitrary number of items (including none).

sequence ::= "(" *item** ")"
R = *C* = "(" *R*<*item**>(*E*) ")"
B = *B*<*item**>(*E*)
T = *L* = L_S = L_t = NIL

Parentheses group a sequence of items as a single value; bindings in the sequence affect the environment of items to the right in the containing node, but labels are disallowed. Parentheses may also be used to override the right-to-left evaluation of arithmetic operators; an operand sequence must reduce to a single numeric value.

binding ::= *name "_" rhs*
R = NIL
B = *localBind*(*name*, *R<rhs>*(*E*), *E*)
where
localBind(*id*, *value*, *E*) = [*E* | *id* _ *value*]
localBind(*id* "." *name*, *value*, *E*) = [*E* | *id* _ *localBind*(*name*, *value*, *valOf*(*id*, *E*))]

This adds a single binding to *E*; bindings have no other "side effects" and no value.

binding ::= *universal ":="* *rhs*
binding ::= *name ":="* *rhs*
R = NIL
B = *globalBind*(*name*, *R<rhs>*(*E*), *E*)
where
globalBind(*name*, *value*, *E*) = if
locVal("OUTER", *E*)=NIL then *localBind*(*name, value, E*)
else [*E* | "OUTER" _ *globalBind*(*name*, *value*, *locVal*("OUTER", *E*))]

Each environment, *E*, initially contains only its "inherited" environment (bound to OUTER). Most bindings take place directly in *E*. To allow for "global" bindings, the value of a *globalBind*(*name*, *R<rhs>*(*E*), *E*) will change *E* by rebinding *id* in the outermost environment X (reached in the semantics by following the OUTER path from *E* until the outermost one is reached; if we started in a nodal environment, this will be X).

Note that a global binding to some variable b does not guarantee that using b in a rhs context will result in accessing the global b because a local binding to b may intervene.

Note that in a context such as [| a := 7], the effect of the above semantics is the same as [| a _ 7].

binding ::= *name mode op term*
= *<name mode name op term>*

This is just a convenient piece of syntactic sugar for the common case of updating a binding.

rhs ::= "'" *item** "'"
R = *item**

If the *rhs* of a binding is surrounded by single quotes, it will be evaluated in the environments where the name is invoked, rather than the environment in which the binding is made.

rhs ::= "[|" *binding** "]"
R = [*B<binding*>*([NULL | "OUTER" "=" *E*]) | "OUTER" "=" NULL]

This creates a new environment value that may be used much like a record.

rhs ::= "[" *invocation* "|" *binding** "]"
R =[*B<binding*>*([*R<invocation>*(*E*) | "OUTER" "=" *E*]) | "OUTER" "=" NULL]

This creates a new environment value that is an extension of an existing one.

tag ::= *universal* "$"
R = *R<valOf*(*universal*, *E*)*>*(*E*)
B = *B<valOf*(*universal*, *E*)*>*(*E*)
T = *universal*
C = *L* = L_S = L_t = NIL

This gives the containing node the property denoted by the universal and also invokes the

universal in the outermost environment (if it is not bound there, NIL will be produced, which contributes nothing to *R*).

link ::= "LINKS" *id*
R = "LINKS" *id*
L = *id*
B = *E*
C = *T* = L_S = L_t = NIL

This defines the scope of the set of links whose "main" component is *id*.

A label *N*: on a node makes that node a "target" of the link *N* (and its prefixes); a reference ^*N* makes it a "source." The "main" identifier of a link must be declared (using LINKS *id*) at the root of a subtree containing all its sources and targets. The link represents a set of directed arcs, one from each of its sources to each of its targets. Multiple target labels make a node the target of multiple links. A target label that appears only on a single node places it in a singleton set, i.e., identifies it uniquely.

link ::= "^" *name*
R = "^" *name*
L_S = *name*
B = *E*
C = *T* = *L* = L_t = NIL

This identifies the containing node as a "source" of the *link* name.

link ::= *name* ":"
R = *name* ":"
L_t = *prefixes*(*name*)
B = *E*
C = *T* = *L* = L_S = NIL

where

prefixes(*id*) = *id*
prefixes(*name* "." *id*) = *name* "." *id prefixes*(*name*)

This identifies the containing node as a "target" of each of the links that is a prefix of *name*.

C.5. Discussion

Each script is evaluated in the context of an initial environment, *X*, which can contain attributes global to all scripts, attributes that specify values for system-specific identifiers, and in which all global bindings are made.

Each environment, *E*, initially contains only its "inherited" environment (bound to the OUTER). Most bindings take place directly in *E*. To allow for more persistent bindings, the value of a *bind*(*id*, ":=", *val*, *E*) will change *E* by rebinding *id* in *X*. For the root node of a script, OUTER = *X*.

If the right-hand side of a binding is surrounded by single quotes, it will be evaluated in the environments where the name is invoked, rather than the environment in which the binding is made.

When an *id* is referred to and *locBinding*(*id*, *E*)=NONE, then the value is sought recursively in

locVal("OUTER"). The *X* environment binds each id to the "universal" name which is its uppercase equivalent (e.g., the universal for iDentiFieR is IDENTIFIER).

Nodes are delimited by brackets. The contents of each node are implicitly prefixed by Sub, which will generally be bound in the containing environment to a quoted expression performing some bindings, and perhaps supplying some labels (tags and links).

Parentheses are used to delimit sequence values. Square brackets are used to delimit the argument list of an operator application and to denote environment constructors, which behave much like records.

Expressions involving the four infix ops (+, -, *, /) are evaluated right-to-left (a la APL); since we expect expressions to be short, we have not imposed precedence rules.

The notation for selections (conditionals) is borrowed from Algol 68:

(<test> | <true part> | <false part>)

This is consistent with our principles of using balanced brackets for compound constructions and avoiding syntactically reserved words; the true part and false part may each contain an arbitrary number of items (including none).

A label N: on a node makes that node a "target" of the link N (and its prefixes); a reference ^N makes it a "source." The "main" identifier of a link must be declared (using LINKS id) at the root of a subtree containing all its sources and targets. The link represents a set of directed arcs, one from each of its sources to each of its targets. Multiple target labels make a node the target of multiple links. A target label that appears only on a single node places it in a singleton set, i.e., identifies it uniquely.

C.6. Grammatical feature X Semantic function matrix

LEGEND:
- Semantic function produces NIL or E or does not apply.
+ Non-trivial semantic equation.
= For R: passes value unchanged; for C: value same as R.

FEATURES:	FUNCTIONS: R	C	B	T	L	L_s	L_t
term ::= primary op term	+	=	-	-	-	-	-
primary ::= literal	=	=	-	-	-	-	-
invocation ::= id	+	-	+	-	-	-	-
invocation ::= name "." id	+	-	+	-	-	-	-
universal ::= name "$"	=	=	-	-	-	-	-
application ::= invocation "[" item* "]"	+	-	-	-	-	-	-
selection ::= "(" term "|" item1* "|" item2* ")"	+	-	+	-	-	-	-
node ::= "{" item* "}"	+	=	+	-	-	-	-
sequence ::= "(" (value | binding)* ")"	+	=	+	-	-	-	-
item* ::= item1 item*	+	+	+	+	+	+	+
binding ::= name mode rhs	-	-	+	-	-	-	-
rhs ::= "'" item* "'"	+	-	-	-	-	-	-
rhs ::= "[|" binding* "]"	+	-	-	-	-	-	-

rhs ::= "[" invocation "|" binding* "]"	+	-	-	-	-	-	-
tag ::= invocation "%"	+	-	-	+	-	-	-
link ::= "LINKS" id	=	-	-	-	+	-	-
link ::= "^" name	=	-	-	-	-	+	-
link ::= name ":"	=	-	-	-	-	-	+

- Semantic function produces NIL or E or does not apply.
+ Non-trivial semantic equation.
= For R: passes value unchanged; for C: value same as R.

HISTORY LOG

Edited by Mitchell, September 1, 1981 3:12 PM, added first version of glossary

Edited by Mitchell, September 7, 1981 2:11 PM, wrote parts of introduction

Edited by Mitchell, September 10, 1981 10:14 AM, added Tab def to Star property sheets

Edited by Mitchell, September 14, 1981 9:54 AM, renumbered chapters and did minor edits

Edited by Horning, May 4, 1982 5:16 PM, Fold in Truth Copy changes, add Appendix B

Edited by Mitchell, May 10, 1982 5:40 PM, changed "Interdoc" to "Interscript", "rendering" to "internalizing", and "transcribing" to "externalizing" plus various edits necessitated by these substitutions.

Edited by Mitchell, August 19, 1982 4:55 PM, preparing the *final* version: eliminated const bindings, changed syntax for links, renamed Outermost to be X.