Two Normal Forms for Link-Connector Pairs in NCL 3.0

Guilherme Augusto Ferreira Lima
Department of Informatics
PUC-Rio, Rio de Janeiro, Brazil
glima@inf.puc-rio.br

Luiz Fernando Gomes Soares
Department of Informatics
PUC-Rio, Rio de Janeiro, Brazil
lfgs@inf.puc-rio.br

ABSTRACT

In this paper, we investigate the problem of normal forms for links and connectors in NCL 3.0. We identify two such forms, called the First and Second Normal Forms (NF1 and NF2), in which links and connectors appear in simple terms. We also present normalization procedures (proofs), which show that for every NCL 3.0 program, there is an equivalent program in each of the forms. The mere existence of NF1 and NF2 makes the semantic analysis of programs simpler. Moreover, the symmetry exhibited by these forms suggests that the same principle of arbitrarily ordered evaluation underlies both the evaluation of link conditions and the execution of non-sequential compound actions.

Categories and Subject Descriptors: I.7.2 [Document Preparation]: Languages and systems

General Terms: Languages, Theory

Keywords: Connectors; links; NCL; Nested Context Language; normal form; semantics

1. INTRODUCTION

Links and connectors are the primary constructs for media synchronization in NCL 3.0 [1, 5]. Links define causal relationships between events in a presentation; connectors define reusable templates for links. Every link-connector pair[1] has two parts: condition and action. The condition specifies the events to be waited together with a predicate, or "assessment statement" in NCL terminology, to be evaluated at each occurrence of the former. The action specifies the events to be generated whenever the condition is satisfied, i.e., whenever the awaited events occur and, simultaneously, the associated predicate evaluates to true.

[1]We shall use the term "link-connector pair" to refer to the causal relationship denoted by a link together with its connector. This terminology is justified because, in terms of program behavior, these constructs are virtually indissociable, with the connector being an integral part of the link definition.

There are several ways in which one can restrict the form of links and connectors without affecting the expressiveness of NCL 3.0.[2] In this paper, we present two such restricted, or *normal*, forms with some interesting properties. More specifically, we show that for every NCL program X there is an equivalent program X' such that, for each link-connector pair ℓ in X', the following properties hold.

1. The condition of ℓ consists of an atomic condition together with a predicate.
2. The action of ℓ is either atomic or is the sequential composition of two atomic actions.

Properties (1) and (2) correspond to what we call the *First Normal Form* (NF1) for NCL programs. We also define a *Second Normal Form* (NF2), which consists of property (1) together with the following properties.

3. The action of ℓ is either of the same format as that of (2) or is the arbitrarily ordered, viz., parallel, composition of $n \geq 2$ actions of the same format as that of (2).
4. For every ℓ' in X' with $\ell' \neq \ell$, the condition of ℓ' is different from that of ℓ.

The implication of property (1) to works like [2, 3, 4, 6, 7, 8], which analyze the behavior of links and connectors in NCL, is immediate: one is freed from the intricate cases of link-connector pairs containing multiple conditions. Similar arguments apply to properties (2)–(4). However, properties (2)–(4) also have a subtler, more profound implication. The manifest symmetry between (2) and (3)–(4) suggests that the principle of arbitrarily ordered evaluation involved in the evaluation of conditions of distinct links is equivalent to that involved in the execution of non-sequential, or *parallel*[3], actions; an observation that might prove relevant for future semantic investigations.

We can convert any NCL program into an equivalent program in NF1 or NF2. The conversion, or *normalization*, procedure consists of several stages, or *reductions*, wherein link-connector pairs that violate the properties of the particular normal form are replaced by equivalent pairs that are satisfactory, or that are closer to be satisfactory than the original ones. The reductions are applied repeatedly until all link-connector pairs become satisfactory.

Although we try to be rigorous in justifying correctness of each reduction, our approach to language semantics is essentially informal. We consider two NCL programs X and X' *equivalent* iff (if, and only if) both define the same set of possible presentations. In particular, if X and X' are deterministic programs, then each defines a single presentation. These are considered equivalent iff, for every

[2]From now on, we shall simply speak of NCL with the suffix "3.0" being tacitly understood.
[3]We shall use the term "parallel" in the particular sense defined by the NCL specification [1, p. 61][5, p. 40], viz., that of evaluation in an arbitrary order, and not in the sense of concurrent evaluation, which is its usual meaning.

input event e, at any given time, the result of applying e to each presentation is exactly the same. By "result," we mean what users see on screen and hear from speakers. We extend this definition to nondeterministic programs X and X' by requiring that the set of results induced by e on X, at any given time, be equal to the set induced by e on X' at that same time, and vice versa.

To simplify the definition of the reductions we introduce, in Section 2, the abstract syntax ABS for the unified representation of link-connector pairs in NCL programs. ABS hides away some idiosyncrasies of the concrete syntax of NCL, e.g., the distinction between links and connectors, *n*-ary compositions, etc., that would complicate the definitions and proofs presented in Sections 3 and 4. Nevertheless, the mapping of ABS programs into equivalent NCL programs is straightforward. The same applies to the normalization procedures, which we define only for ABS programs. The mapping of these procedures into equivalent procedures that operate on NCL programs follows directly from the previous mapping.

The rest of the paper is organized as follows. Section 2 presents the syntax and semantics of ABS, and discusses the mapping of its programs into equivalent NCL programs. Sections 3 and 4 present, respectively, the First and Second Normal Form theorems for ABS programs, with their proofs corresponding to what we termed the "normalization procedures." Finally, Section 5 concludes the paper.

2. THE ABSTRACT SYNTAX

In this section, we introduce ABS, a simple language for the unified representation of link-connector pairs in NCL programs. First, we define the syntax of ABS, i.e., the structure of the expressions which we regard as well-formed programs. Then, we present the intended interpretation for these programs. Finally, we discuss the mapping of NCL programs into equivalent ABS programs.

ABS has five main syntactic sets: programs **S**, links **L**, conditions **C**, predicates **P**, and actions **A**. The structure of the members of these sets is given by the following BNF-like grammar, where '::=' read as "can be," '|' read as "or," ε stands for the empty string, and the metavariables S, L, C, P, and A, with or without super or subscripts, are assumed to range over the sets **S**, **L**, **C**, **P**, and **A**, respectively.

$$\begin{aligned}
S &::= LS_0 \mid \varepsilon \\
L &::= C \rightarrow A \\
C &::= (C_0 \wedge C_1 \ ?\ P) \mid (C_0 \vee C_1 \ ?\ P) \mid (c \ ?\ P) \\
P &::= (P_0 \wedge P_1) \mid (P_0 \vee P_1) \mid (\neg P_0) \mid p \\
A &::= (A_0, A_1) \mid (A_0 \parallel A_1) \mid a
\end{aligned}$$

The definition of conditions (C), predicates (P), and actions (A) contain the metavariables c, p, and a. These are assumed to range over the primitive syntactic sets **c** of atomic conditions, **p** of atomic predicates, and **a** of atomic actions, respectively. The partial structure of the members of these primitive sets is given below.

$$\begin{aligned}
c &::= u.v \mid \cdots \\
p &::= u.v_0 = v_1 \mid \cdots \\
a &::= u.v_0 := v_1 \mid \cdots
\end{aligned}$$

Here '$\cdots$' stands for the omitted forms, i.e., those whose structure is of no particular interest to us, and the metavariables u and v, with or without subscripts, are assumed to range over the sets **u** of component (i.e., media, context, or switch) identifiers and **v** of arbitrary strings, respectively. We assume that the structure of the members of **u** and **v** is given.

By the above definition, an ABS program S is simply a finite list of links $L_1, L_2, \ldots, L_n$ with $n \geq 0$. Each link L_i, for $1 \leq i \leq n$, is of the form $C \rightarrow A$ and establishes that whenever condition C is satisfied, action A is executed. Condition C is either the conjunction ($\wedge$) or disjunction ($\vee$) of two other conditions C_0 and C_1 associated with a predicate P, viz., the expression at the right-hand side of symbol '?', or is an atomic condition c also associated with a predicate P. Predicate P, in turn, is either the conjunction ($\wedge$) or disjunction ($\vee$) of two other predicates P_0 and P_1, the negation ($\neg$) of another predicate P_0, or is an atomic predicate p. Finally, action A is either the sequential (,) or parallel ($\parallel$) composition of two other actions A_0 and A_1, or is an atomic action a.

Moreover, the atomic condition c specifies a single event to be waited (e.g., the pressing of a button), the atomic predicate p specifies a Boolean test involving the equality or inequality of properties (i.e., NCL `<property>` elements) and values (i.e., arbitrary strings), and the atomic action a specifies an event to be generated (e.g., the starting of the presentation of some media object). The particular forms of c, p, and a that were singled out are to be interpreted as follows: The atomic condition $u.v$ is satisfied whenever some value is stored in property v of component u, the atomic predicate $u.v_0 = v_1$ evaluates to true if the content of property v_0 of component u is equal to the string v_1, and the atomic action $u.v_0 := v_1$, if executed, stores the string v_1 into property v_0 of component u.

The mapping of an arbitrary NCL program X into an equivalent ABS program is direct if X satisfies the following restrictions.

1. Each connector of X is referenced by exactly one link.
2. The connectors and links of X contain no link, bind, or connector parameters, i.e., no `<linkParam>`, `<bindParam>`, or `<connectorParam>` elements.
3. The compound conditions and compound actions of all connectors of X have no *delay* attribute.
4. The simple conditions and simple actions of all connectors of X are referenced by exactly one bind (i.e., `<bind>` element) in the associated links.
5. The compound actions and compound statements of all connectors of X are binary and its compound conditions are either binary or ternary, and have exactly one child (assessment or compound) statement.

If X satisfies the above restrictions, we can build an equivalent ABS program by mapping each link-connector pair ℓ of X into an equivalent element of **L**. The mapping is defined by recursion as follows: For each ℓ, we map each of its `<compoundCondition>`, `<compoundStatement>`, and `<compoundAction>` elements into equivalent members of **C**, **P**, and **A**, respectively, and, at the basis of the recursion, we map each of its `<simpleCondition>`, `<assessmentStatement>`, and `<simpleAction>` elements, together with the associated `<role>` elements, into equivalent members of **c**, **p**, and **a**, respectively.

If, however, X does not satisfy restrictions (1)–(5), we first need to convert it into an equivalent satisfactory NCL program, which is then mapped into ABS. Since this extra conversion, or *prenormalization*, stage is not particularly enlightening we shall only discuss it briefly.[4] The prenormalization procedure consists, basically, of the consecutive application of steps (1)–(5) below.[5]

1. Make a copy of each connector that is referenced by more than one link and update the links to point to different copies.

[4]This stage is implemented by prenormalization module of the DietNCL conversion tool, cf. `http://www.telemidia.puc-rio.br/~gflima`.

[5]More precisely, for each step $1 \leq i \leq 5$ and each program X, if we apply the prenormalization steps (1)–(i) to X we get an equivalent program X' that satisfies restriction (i).

2. Replace, in each connector, the value of the parameters defined in the associated link.
3. Add the *delay* attribute of each compound condition (or action) to the *delay* of its components. Then remove the *delay* attribute from the parent composition. Repeat this procedure until all compositions have no *delay* attribute.
4. Replace each simple condition (or action) that is referenced by more than one bind by an equivalent compound condition (or action) and update the *role* of the referring binds to point to the new simple conditions (or actions). Repeat this procedure until every simple condition (or action) is referenced by exactly one bind.
5. Replace all simple conditions by a binary compound condition containing the original simple condition together with a tautological assessment statement. Then break all n-ary compound conditions, with $n \geq 3$, into an equivalent chain of binary ones, adding vacuous, tautological statements whenever necessary. Finally, repeat the breakage procedure for compound actions and compound statements.

3. FIRST NORMAL FORM

We now undertake the task of stating and proving the first normal form theorem for ABS programs. First, however, we need to introduce some notation.

Let e_0 and e_1 be elements of the same syntactic set. Then we write $e_0 \equiv e_1$ iff e_0 is identical to e_1, i.e., iff they have the same parse tree.

The *condition degree* d of a link $L \equiv C \to A$ is defined inductively by the following clauses.

1. If $C \equiv (c \mathbin{?} P)$ then $d = 0$.
2. If $C \equiv (C_0 \wedge C_1 \mathbin{?} P)$ or $C \equiv (C_0 \vee C_1 \mathbin{?} P)$, and the condition degrees of C_0 and C_1 are, respectively, d_0 and d_1, then $d = d_0 + d_1 + 1$.

Similarly, the *action degree* d of a link $L \equiv C \to A$ is defined by:

1. If $A \equiv a$ or $A \equiv (a_0, a_1)$ then $d = 0$.
2. If $A \equiv (A_0, A_1)$, with $A_i \not\equiv a$ for some $i = 0$ or $i = 1$, or $A \equiv (A_0 \parallel A_1)$, and the action degrees of A_0 and A_1 are, respectively, d_0 and d_1, then $d = d_0 + d_1 + 1$.

We shall use the letter 'F', with or without subscripts, to represent *flags*: specially crafted properties (i.e., `<property>` elements) that function as private variables. We use flags to simulate the behavior of the particular construction we are trying to eliminate. Flags may appear in conditions, predicates, and actions. E.g., link $(F \mathbin{?} P) \to A$ establishes that whenever some value is stored in flag F and, simultaneously, predicate P holds (i.e., evaluates to true), then action A is executed; link $(C \mathbin{?} F = 1) \to A$ establishes that whenever C is satisfied and, simultaneously, the content of flag F is equal to 1, then A is executed; and link $C \to F := 1$ establishes that whenever C is satisfied, then 1 is stored in flag F. Initially, every flag is assumed to contain 0.

We now prove a basic lemma about the elimination of non-atomic conditions.

Lemma 1. *For any ABS program S there is an equivalent program S' such that each link L of S' has condition degree zero, i.e.,*

$$L \equiv (c \mathbin{?} P) \to A,$$

for some atomic condition c, predicate P, and action A.

Proof. Let S be an arbitrary ABS program, and let L be some link of S with condition degree $d_c > 0$. Then either

$$L \equiv ((C_0 \mathbin{?} P) \vee (C_1 \mathbin{?} Q) \mathbin{?} R) \to A \tag{1}$$

or

$$L \equiv ((C_0 \mathbin{?} P) \wedge (C_1 \mathbin{?} Q) \mathbin{?} R) \to A, \tag{2}$$

for some conditions C_0 and C_1, predicates P, Q, and R, and action A.

In case (1), action A is executed if any of the conditions $(C_0 \mathbin{?} P)$ or $(C_1 \mathbin{?} Q)$ are satisfied and if, simultaneously, predicate R holds. We remove link L from S by transforming it into the links

$$(C_0 \mathbin{?} P \wedge R) \to A$$
$$(C_1 \mathbin{?} Q \wedge R) \to A,$$

which execute A if C_0 is satisfied and $P \wedge R$ holds, or C_1 is satisfied and $Q \wedge R$ holds, or both. Thus the transformation maintains the original behavior.

In case (2), action A is executed immediately after conditions $(C_0 \mathbin{?} P)$ and $(C_1 \mathbin{?} Q)$ are satisfied (in fact, just after the last one of them is satisfied) and only if, at that moment, predicate R holds. In this case, we remove link L from S by transforming it into the links

$$\begin{gathered}(C_0 \mathbin{?} P \wedge R) \to F_{C_0} := 1\\ (C_1 \mathbin{?} Q \wedge R) \to F_{C_1} := 1\\ (F_{C_0} \mathbin{?} F_{C_1} = 1) \to (F_{(C_0 \wedge C_1)} := v, A)\\ (F_{C_1} \mathbin{?} F_{C_0} = 1) \to (F_{(C_0 \wedge C_1)} := v, A)\\ (F_{(C_0 \wedge C_1)} \mathbin{?} \top) \to (F_{C_0} := 0, F_{C_1} := 0),\end{gathered}$$

where $\top$ denotes some tautological predicate, e.g., $(p \vee \neg p)$, and v denotes some arbitrary string. These links execute A immediately after condition $(C_0 \mathbin{?} P \wedge R)$ is satisfied if $(C_1 \mathbin{?} Q \wedge R)$ was satisfied earlier, or after $(C_1 \mathbin{?} Q \wedge R)$ if $(C_0 \mathbin{?} P \wedge R)$ was satisfied earlier. Thus the transformation maintains the original behavior.

In either case, the links replaced for L have condition degree less than d_c. Therefore, by successively repeating the transformations, we obtain an equivalent program S' in which all links have condition degree zero. □

We proceed to prove the main result of this section.

Theorem 1 (First Normal Form, or NF1). *For any ABS program S there is an equivalent program S' such that each link L of S' has condition and action degrees zero, i.e.,*

$$L \equiv (c \mathbin{?} P) \to a_0 \qquad \text{or} \qquad L \equiv (c \mathbin{?} P) \to (a_0, a_1),$$

for some atomic condition c, predicate P, and atomic actions a, a_0, and a_1.

Proof. Let S'' be the result of applying Lemma 1 to S, and let L be some link of S'' with an action degree $d_a > 0$. Then either

$$L \equiv C \to (A_0 \parallel A_1) \qquad \text{or} \qquad L \equiv C \to (A_0, A_1),$$

for some condition $C \equiv (c \mathbin{?} P)$ and actions A_0 and A_1.

In the first case, if condition C is satisfied then actions A_0 and A_1 are executed in "parallel," i.e., in an arbitrary order. We remove link L from S'' by transforming it into the links

$$C \to A_0$$
$$C \to A_1,$$

which clearly maintain the original behavior, since, in NCL, the order of evaluation of links is also arbitrary.

In the second case, if condition C is satisfied then actions A_0 and A_1 are executed in sequence, i.e., A_0 is executed before A_1. There are the following three possibilities. (Note that all of them guarantee that the links replaced for L execute A_0 before A_1 whenever C is satisfied.)

1. If $A_1 \not\equiv a$, for any atomic action a, we replace L by

$$C \to (A_0, F_{A_0} := v)$$
$$(F_{A_0} \ ?\ \top) \to A_1.$$

2. If $A_0 \equiv (A'_0, A''_0)$, for some A'_0 and A''_0, we replace L by

$$C \to (A'_0, F_{A'_0} := v)$$
$$(F_{A'_0} \ ?\ \top) \to (A''_0, A_1).$$

3. If $A_0 \equiv (A'_0 \parallel A''_0)$, for some A'_0 and A''_0, we replace L by

$$C \to (A'_0, F_{A'_0} := 1)$$
$$C \to (A''_0, F_{A''_0} := 1)$$
$$(F_{A'_0} \ ?\ F_{A''_0} = 1) \to (F_{(A'_0 \| A''_0)} := v, A_1)$$
$$(F_{A''_0} \ ?\ F_{A'_0} = 1) \to (F_{(A'_0 \| A''_0)} := v, A_1)$$
$$(F_{(A'_0 \| A''_0)} \ ?\ \top) \to (F_{A'_0} := 0, F_{A''_0} := 0).$$

In any case, the links replaced for L have condition degree zero and have action degree less than d_a. Therefore, by successively repeating the transformations, we obtain an equivalent program S' in which all links have condition and action degrees zero. □

4. SECOND NORMAL FORM

We now turn to the statement and proof of the second normal form theorem for ABS programs. To characterize the structure of actions in this normal form, we introduce the concept of sequential degree.

The *sequential degree* d of a link $L \equiv C \to A$ is defined by:

1. If $A \equiv a$ or $A \equiv (a_0, a_1)$ then $d = 0$.
2. If $A \equiv (A_0 \parallel A_1)$ and the sequential degrees of A_0 and A_1 are, respectively, d_0 and d_1, then $d = d_0 + d_1$.
3. If $A \equiv (A_0, A_1)$, with $A_i \not\equiv a$ for some $i = 0$ or $i = 1$, and the sequential degrees of A_0 and A_1 are, respectively, d_0 and d_1, then $d = d_0 + d_1 + 1$.

The following corollary is a direct consequence of Theorem 1.

Corollary 1. *If S is an ABS program in NF1, then all its links have sequential degree zero.*

Proof. By Theorem 1, every link of S has action degree zero. Thus, by the first clause of the definitions of action and sequential degrees, every link of S has sequential degree zero. □

We proceed to establish the main result of this section.

Theorem 2 (Second Normal Form, or NF2). *For any ABS program S there is an equivalent program S' such that each link L of S' has condition and sequential degrees zero, i.e.,*

$$L \equiv (c \ ?\ P) \to A_0 \quad \text{or} \quad L \equiv (c \ ?\ P) \to (A_1 \parallel A_2 \parallel \cdots \parallel A_n),$$

for some atomic condition c, predicate P, and actions $A_0, A_1, \ldots, A_n$ such that $A_i \equiv a$ or $A_i \equiv (a_0, a_1)$, for $0 \le i \le n$. Moreover, for each pair of distinct links $L' \equiv C' \to A'$ and $L'' \equiv C'' \to A''$ of S',

$$C' \not\equiv C'',$$

i.e., no two links of S' have the same condition.

Proof. Let S'' be the result of applying Theorem 1 to S, and let $L' \equiv C' \to A'$ and $L'' \equiv C'' \to A''$ be links of S'' such that $C' \equiv C''$. Then, by Theorem 1, L' and L'' have condition degree zero, and by Corollary 1, they have sequential degree zero. We remove L' and L'' by transforming them into a single link of the form

$$C' \to (A' \parallel A''), \tag{3}$$

which clearly maintains the original behavior. Moreover, (3) has condition and sequential degrees zero—its condition degree is equal to that of C', which is zero, and its sequential degree is equal to the sum of the degrees of A' and A'', which are also zero. Therefore, by successively repeating the transformations we obtain an equivalent program S' in which all links have condition and sequential degrees zero and such that no two links have the same condition. □

5. CONCLUSION

In this paper, we investigated the problem of normal forms for links and connectors in NCL 3.0. Two such forms, termed the First and Second Normal Forms, were identified and precisely defined. Moreover, we showed that for every NCL 3.0 program there is an equivalent program in each of the forms. We also discussed the apparent duality between these forms, which suggests that the same principle of arbitrarily ordered evaluation underlies both the evaluation of conditions and the execution of non-sequential, "parallel" actions. We hope this result prove useful for future investigations.

A related problem, not addressed in this paper, is the question whether NF1 and NF2 are irreducible. We believe that to be the case, but we still do not have a proof, which might require the formalization of program semantics.

REFERENCES

[1] ABNT NBR 15606-2. Digital Terrestrial TV – Data coding and transmission specification for digital broadcasting – Part 2: Ginga-NCL for fixed and mobile receivers: XML application language for application coding. ABNT, São Paulo, SP, Brazil, November 2007.

[2] Costa, R. M. R., Moreno, M. F., and Soares, L. F. G. Intermedia synchronization management in DTV systems. In *Proceedings of the 8th ACM Symposium on Document Engineering - DocEng'08* (São Paulo, SP, Brazil, September 2008), ACM, New York, NY, USA, pp. 289–297.

[3] dos Santos, J., Braga, C., and Saade, D. C. M. A model-driven approach for the analysis of multimedia documents. In *Proceedings of the Doctoral Symposium of the 5th International Conference on Software Language Engineering - SLE 2012* (Dresden, Germany, September 2012).

[4] Felix, M. F., Haeusler, E. H., and Soares, L. F. G. Validating hypermedia documents: A timed automata approach. Monografias em Ciência da Computação, PUC-Rio, Rio de Janeiro, RJ, Brazil, 2002.

[5] ITU-T Recommendation H.761. Nested Context Language (NCL) and Ginga-NCL for IPTV Services. ITU-T, Geneva, Switzerland, April 2009.

[6] Lima, G. A. F., Soares, L. F. G., Neto, C. S. S., Moreno, M. F., Costa, R. R., and Moreno, M. F. Towards the NCL Raw Profile. In *II Workshop de TV Digital Interativa (WTVDI) - Colocated with ACM WebMedia'10* (Belo Horizonte, MG, Brazil, October 2010).

[7] Picinin, Jr., D., Farines, J.-M., and Koliver, C. An approach to verify live NCL applications. In *Proceedings of the 18th Brazilian Symposium on Multimedia and the Web - WebMedia'12* (São Paulo, SP, Brazil, October 2012), ACM, New York, NY, USA, pp. 223–232.

[8] Yovine, S., Olivero, A., Monteverde, D., Cordoba, G., and Reiter, L. An approach for the verification of the temporal consistency of NCL applications. In *II Workshop de TV Digital Interativa (WTVDI) - Colocated with ACM WebMedia'10* (Belo Horizonte, MG, Brazil, October 2010).