Sufficient Conditions for Well-behaved Adaptive Hypermedia Systems

Hongjing Wu, Paul De Bra

Department of Computing Science
Eindhoven University of Technology
Eindhoven, the Netherlands
email: {hongjing,debra}@win.tue.nl

Abstract. We focus on well-behaved Adaptive Hypermedia Systems, which means the adaptation engine that executes adaptation rules always terminates and produces predictable (confluent) adaptation results. Unfortunately termination and confluence are undecidable in general. In this paper we discuss sufficient conditions to help authors to write adaptation rules that satisfy termination and confluence.

Keywords: adaptive hypermedia, user modeling, production rules.

1. Introduction

Adaptive Hypermedia Systems (or AHS for short) provide automatically personalized access to hypermedia information sources, most often in the form of Websites. Most AHS provide *adaptive navigation support* and *adaptive content*. The link structure or the presentation of link anchors is different for every user. The actual content on information pages is also different for every user. An overview of systems, methods and techniques for adaptive hypermedia can be found in [B96]. We have developed a reference model for the architecture of adaptive hypermedia applications: the **A**daptive **H**ypermedia **A**pplication **M**odel (AHAM) [DHW99]. AHAM describes AHS at an abstract level, using an architecture consisting of three parts:

- a *domain model* (DM) that describes how the information content of the application is structured (using concepts and concept relationships).
- a fine-grained *user model* (UM) that represents a user's preferences, knowledge, goals, navigation history and other relevant aspects.
- an *adaptation model* (AM) consisting of *adaptation rules*. The rules define the process of generating the adaptive presentation and of updating the user model.

This architecture provides a clear separation of concerns when developing an adaptive hypermedia application.

The implementation part of AHS, called *adaptation engine* (AE), is the software that performs the adaptation (described by rules in AM). *Design issues for a general-purpose adaptation engine* (AE) are discussed in earlier paper [WDD01]. We defined a rule language for AHS, AHAM-CA and proposed a static analysis method to decide if for a given DM, UM and AM the AE would always *terminate* and if it would be *confluent* (meaning that the system would generate predictable results). In this paper we discuss how the (sufficient but not necessary) conditions that guarantee *termination* and *confluence* can be relaxed (while remaining sufficient).

2. The Adaptation Rule Language

For lack of space we will not give a complete specification of the syntax of our (abstract) rule language, but illustrate it with an example. For details, see [WDD01]. A rule C→A in AHAM consists of a condition (C) and an action (A). While the properties of the language are independent of the syntax that is used, we use an SQL-like syntax for clarity. An example of an AHAM-CA rule is:

C: **select** C_1.knowledge
where C_1.knowledge ≥ "known"
A: **update** C_2.ready_to_read := **true**
where prerequisite(C_1, C_2) **and**
not exists (**select** C_3
where prerequisite(C_3, C_2) **and**
C_3.knowledge < "known")

This is a *generic* rule, containing concept variables C_1, C_2, and C_3. The language also allows for (more or less) *specific* rules that use concepts instead of concept variables. The example rule says that when the knowledge of concept C_1 changes so that it becomes at least "known" then all concepts C_2 for which C_1 was the last prerequisite that was not yet "known" now become "ready_to_read". In this rule language it is all too easy to write rules that may cause infinite loops or unpredictable results. An example of such a rule is:

C: **select** C_1.attr
where C_1.attr > 0
A: **update** C_2.attr := C_2.attr + 1
where rel(C_1, C_2)

This example also shows that whether or not this rule generates an infinite loop depends on whether the concept relationship "rel" has cycles.

3. Sufficient Conditions for Termination and Confluence in AHS

The AHAM-CA rule language is very powerful, but expressive power always has an impact on system behavior. There is no implicit guarantee that the system is well-behaved. We proposed a static analysis method for termination and confluence [WDD01] for general cases in AHS; the analysis either tells us the rule set is confluent and terminates, or that this can't be determined. This section defines some constraints on AHAM-CA rules that guarantee termination and confluence, while at the same time retaining more freedom for the author to write propagation than the sufficient conditions of [WDD01]. We first define some terms and functions that will be used later on.

Definition 1: *R_i may activate R_j* if the execution of action A_i *can* change the database (DM and UM together) from a state in which condition C_j is false to a state in which C_j is true. *R_i may deactivate R_j* if the execution of action A_i *can* change the database from a state in which C_j is true to a state in which C_j is false.

Definition 2: A rule set *terminates* if the rules cannot activate each other indefinitely.

Definition 3: A *rule execution state* S is a pair (d, R_A), where d is a database state (DM and UM) and $R_A \subseteq$ AM is a set of active rules.

Definition 4: A *rule execution sequence* is a sequence σ consisting of a series of rule execution states linked by (executed) rules. A rule execution sequence is *complete* if the last state is (d,$\emptyset$), i.e., the last state has no active rules. A rule execution sequence is *valid* if it represents a correct execution sequence: only active rules are executed, and pairs of adjacent states properly represent the effect of executing the corresponding rule; for details see [AHW95].

Definition 5: A rule set is *confluent* if, for every initial rule execution state S (produced by an initial database state followed by a set of user modifications), every valid and complete rule execution sequence beginning with S has the same final state.

Definition 6:

1. Let R: C→A. the function *num* for the number of relationships used in the **where** clause of A (A.**where)** (the exact definition is omitted because of limited space).
2. Let R: C→A

(a) S(R) = the set of attributes which are selected in C
(b) U(R) = the set of attributes to which values are assigned in A
(c) E(R) = the set of attributes used in the right-hand side of assignments in A
(d) A *st-rule* (start rule) is a rule that is triggered by external events or internally generated events. Its action only updates the concept selected by its condition. It describes the change of the values inside the same concept. St-rule represents a set of *st-rules*.
(e) A *pr-rule* (propagation rule) is a rule that propagates the changes of values to different concepts through relationships between these concepts. It would be a sign of bad design if rules propagate changes through means other than concept relationships in AHS. Pr-rule represents a set of *pr-rules*.
(f) Pri(R) is the number to represent the priority of the execution order of rule R.
(g) AM(rel)⊆Pr-rule is the set of rules which propagate their change "through" the relationship type rel.

Now we study constraints that guarantee that the execution of a set of rules terminates and is confluent, and that still give authors a certain expressive power to write rules with propagation. The first few constraints show that a straightforward approach leads to very strict constraints that do not give authors enough freedom.

Constraint 1: $\forall R_i, R_j \in$ AM: $S(R_i) \cap U(R_j) = \emptyset$.
This constraint means that rules are not allowed to trigger each other.

Theorem 1: A rule set AM satisfying Constraint 1 terminates.
We omit the (easy) proof. This constraint is very strict as it prohibits propagation. However, it is not yet sufficient to guarantee confluence.

Constraint 2: $\forall R_i, R_j \in$ AM:

1. R_i is *independent* from R_j: $(S(R_i) \cup U(R_i) \cup E(R_i)) \cap U(R_j) = \emptyset$.
2. R_i is *self-independent*: $(S(R_i) \cup E(R_i)) \cap U(R_i) = \emptyset$.

This constraint means rules are not allowed to affect (activate or deactivate) each other or themselves and rule execution order won't affect the final result.

Theorem 2: A rule set AM satisfying Constraint 2 terminates and is confluent.
This (easy) proof is also omitted.

While Constraint 1 only guarantees termination, constraint 2 is a sufficient condition for termination *and* confluence. The computational complexity of the algorithm to verify these constraints is $O(N^2xM^2)$, where N is the number of rules and M is the number of attributes. These constraints are very strict in the sense that it is impossible to describe any propagation (a rule that activates other rules). We define Constraints 3-7 (and 7') to give more expressive freedom to authors.

Constraint 3: AM = St-rule$\cup$Pr-rule, $\forall R_i \in$ St-rule, $\forall R_j \in$ Pr-rule: $Pri(R_i) > Pri(R_j)$.

This is a general constraint for rules to be semantically correct in AHS. It means that the set of rules consists of start rules and propagation rules. This constraint also describes that in each phase of the transition, the start rules execute before the propagation. Priorities help but are not enough to guarantee the termination or confluence.

Constraint 4: The "graph" for every type of concept relationship (except hyperlinks) is acyclic.

Hyperlinks are not used for propagating user-model changes, so they may have cycles. Other relationships are used to propagate the changes between different concepts. It is often not a good design if the relationship type is cyclic, because then the change propagation may never stop. (For instance, cycles in prerequisite relationships do not make sense.) But even with acyclic relationship types, relationships of different types can still interact and cause non-termination that way.

Constraint 5: $\forall rel_1, rel_2 \in$ DM-rel, $rel_1 \neq rel_2$:
$\forall R_i \in AM(rel_1), \forall R_j \in AM(rel_2): U(R_i) \cap S(R_j) = \varnothing$.

This constraint means that rules that use a different type of relationships cannot activate each other.

Theorem 3: A rule set AM terminates if it satisfies Constraints 3-5.
Proof (sketch): A rule set AM consists of a finite number of st-rules and pr-rules. The st-rules won't trigger each other; they are trigged by external and internal events. The st-rules may trigger the pr-rules, and the pr-rules may also trigger pr-rules. The propagation for rules that use a relationship type always terminates because the relationship graph is a DAG. And rules that use a different type of relationships cannot trigger each other, so different DAGs cannot be combined to form a cycle.

Constraint 6: $\forall rel \in$ DM-rel: $\forall R_i, R_j \in AM(rel), R_i \neq R_j: U(R_i) \cap U(R_j) = \varnothing$.

This constraint says that every pair of rules containing the same relationship type updates disjoint sets of attributes. In a simple propagation case every attribute is assigned to only once per transition.

Definition 7: $\forall rel_1, rel_2 \in$ DM-rel:
Independent(rel_1, rel_2) holds if $\forall R_i \in AM(rel_1), \forall R_j \in AM(rel_2), R_i \neq R_j$:

$(S(R_i) \cup U(R_i) \cup E(R_i)) \cap U(R_j) = \varnothing$ **and** $(S(R_j) \cup U(R_j) \cup E(R_j)) \cap U(R_i) = \varnothing$

This definition says that all relationship types are independent; the execution order of rules using different relationship types doesn't matter to the final result.

Constraint 7: $\forall rel_1, rel_2 \in$ DM-rel, $rel_1 \neq rel_2$, *Independent(rel_1, rel_2)* holds.

Constraint 7': $\forall R \in AM$, $R:C \rightarrow A$: *num*(A.**where**) ≤ 1 **and**
$\forall rel_1, rel_2 \in$ DM-rel: $(\forall R_i \in AM(rel_1), \forall R_j \in AM(rel_2), rel_1 \neq rel_2: Pri(R_i) > Pri(R_j))$ **or**
$(\forall R_i \in AM(rel_1), \forall R_j \in AM(rel_2), rel_1 \neq rel_2: Pri(R_i) < Pri(R_j))$

This constraint means that each pr-rule has at most one relationship, and the propagation order through all relationship graphs is pre-defined. Constraint 7 needs to calculate many attribute sets, and in most cases we need apply different relationships separately, so it is more natural to just define some execution order for them. We can then use Constraint 7' to replace Constraint 7.

Theorem 4: A rule set AM is confluent if it satisfies Constraint 3-7 (or 7').
Proof (sketch): Constraints 3-5 guarantee AM to terminate, with Constraint 6 AM becomes confluent for the propagation through each relationship graph. Furthermore, with Constraint 7 or Constraint 7' AM becomes confluent for propagation through all relationship graphs.

Constraint 3-7 (or 7') are easy to understand for the author and can be used in most common AHS. Constraint 3 is easy to verify, while Constraint 4 and Constraint 7' rely on the DM alone and are known before analyzing a set of rules. As the number of relationship types is small, the time needed for the algorithm to verify Constraint 3-7 (or 7') is similar to the one to verify Constraint 2.

3. Conclusions

In this paper we proposed some constraints on adaptation rules to obtain sufficient conditions that guarantee termination and confluence for AHS. Checking these constraints has a much lower computational complexity than the static analysis method proposed in [WDD01]. Imposing Constraints 3-7 (or 7') still allows an author to write propagating adaptation rules in most common AHS.

References

[AHW95] Aiken, A., Widom, J., Hellerstein, J.M., "Static Analysis Techniques for Predicting the Behavior of Database Production Rules". ACM Transactions on Database Systems, Vol. 20, nr. 1, pp. 3-41, 1995.

[B96] Brusilovsky, P., "Methods and Techniques of Adaptive Hypermedia". User Modeling and User-Adapted Interaction, 6, pp. 87-129, 1996. (Reprinted in Adaptive Hypertext and Hypermedia, Kluwer Academic Publishers, pp. 1-43, 1998.)

[DHW99] De Bra, P., Houben, G.J., Wu, H., "AHAM: A Dexter-based Reference Model for Adaptive Hypermedia". Proceedings of ACM Hypertext'99, Darmstadt, pp. 147-156, 1999.

[WDD01] Wu, H., De Kort, E., De Bra, P., "Design Issues for General Purpose Adaptive Hypermedia Systems". Proceedings of the 12th ACM Conference on Hypertext and Hypermedia, Arhus, Denmark, 2001 (to appear).