Experience-Based Deductive Learning

Joongmin Choi and Stuart C. Shapiro
Department of Computer Science
State University of New York at Buffalo
Buffalo, New York 14260

Abstract
A method of deductive learning is proposed to control deductive inference. Our goal is to improve problem solving time by experience, when that experience monotonically adds knowledge to the knowledge base. Accumulating and exploiting experience are done by the schemes of knowledge migration and knowledge shadowing. Knowledge migration generates specific (migrated) rules from general (migrating) rules, and accumulates deduction experience represented by specificity relationships between migrating and migrated rules. Knowledge shadowing recognizes rule redundancies during a deduction, and prunes deduction branches activated from redundant rules. The major contribution of this work is to improve a system's performance especially deduction speed over time for similar problems, even though the search space (or branching factor) is increased by monotonically adding derived knowledge to the knowledge base that creates redundancies.

2 Knowledge Migration
Knowledge migration is a process of acquiring specific rules from general rules, and is defined as below.

Definition Let \( G = (A, C) \) be a nested rule, where \( A = \{A_1, A_2, \ldots, A_n\} \) is a set of antecedents, \( C = \{C_1, C_2, \ldots, C_m\} \) is a set of consequents, and each \( C_j(1 \leq j \leq m) \) is a rule. Knowledge migration is defined as a process of acquiring a set of rules \( S = \{S_1, S_2, \ldots, S_m\} \) from \( G \) during a deduction when (1) \( G \) is involved in the deduction, and (2) there is a substitution \( \sigma \), called a migrating substitution, such that \( A_i \sigma(1 \leq i \leq n) \) is in the knowledge base, and \( S_j = C_j \sigma(1 \leq j \leq m) \).

As an example, consider a knowledge base containing a general transitive rule \( R1 = \forall R [\text{trans}(R) \rightarrow \forall x, y, z [R(x, y) & R(y, z) \rightarrow R(x, z)]] \), and facts \( \text{trans}(on), \text{on}(a, b), \text{on}(b, c), \text{on}(c, d) \). A deduction of \( \text{on}(a, c) \) from this knowledge base activates \( R1 \), and generates a specific rule \( R2 = \forall x, y, z [\text{on}(x, y) & \text{on}(y, z) \rightarrow \text{on}(x, z)] \) from \( R1 \) with a migrating substitution \( \sigma = \{\text{on}(x)\} \), since \( \text{trans}(R) \sigma = \text{trans}(on) \) is in the knowledge base. This migration process produces deduction experience that is represented by the specificity relationship between the migrating rule and the migrated rule. The details of the representation are described in the next section.

3 Knowledge Shadowing
Knowledge shadowing is a way of exploiting deduction experience acquired in previous inferences to make subsequent similar deductions more efficient. The main job of knowledge shadowing is to recognize unnecessary and redundant rules and to block them...
from the inference by investigating the specificity relationships between rules that are represented in deduction experience.

As an example, consider the example shown in the previous section. After deducing \( on(a,c), R_2 \) and \( on(a,c) \) are added to the knowledge base. Now, another deduction of \( on(b,d) \) from the changed knowledge base proceeds in two branches since both \( R_1 \) and \( R_2 \) are applicable. However, by recognizing that \( R_2 \) is more specific than \( R_1 \), the branch from \( R_1 \) can be regarded as redundant in the sense that \( on(b,d) \) can be derived solely by the branch from \( R_2 \) with fewer steps. As a result, the branch activated from \( R_1 \) is pruned.

Knowledge shadowing is made possible by three ways of representing experience, i.e., by using instance sets, by using origin sets, and by using common instances.

In the first method, an instance set \( T^R \) is maintained for each rule \( R \) to memorize migrated instances. A migrated instance is represented by a pair \((r, \sigma)\), where \( r \) is a rule migrated from \( R \) by a migrating substitution \( \sigma \). For example, after the migration from \( R_1 \) occurs during the derivations of \( on(a,c) \), \( T^{R_1} \) becomes \( \{ \{ R_2, \{ on/R \} \} \} \), where \( R_1_{on} = \{ x/z, y/z \} \) \( \& \) \( R_2(y, z) \rightarrow R(x, z) \) is the consequent of \( R_1 \). Knowledge shadowing by using instance sets can be accomplished by the following principle.

**Shadowing Principle 1** Let \( R_1, R_2, \ldots, R_k \) be rules that are all applicable by a query \( q \) at some point during a deduction. This implies the existence of substitutions \( \phi_1, \phi_2, \ldots, \phi_k \) of \( R_1, \ldots, R_k \), respectively. Then, a deduction branch activated by \( R_i \) is shadowed from the inference when there is an instance \((R_j, \sigma) \in T^R \) such that \( 1 \leq j \leq k, j \neq i \), and \( \phi_i \supseteq \sigma \).

In the transitive example, the query \( on(b,d) \) makes both \( R_1 \) and \( R_2 \) applicable but shadows \( R_1 \) by the principle 1, since the pattern matching between the query \( on(b,d) \) and \( R(x,z) \), which is the consequent of \( R_1 \), produces a unifier \( \phi = \{ on/R, b/z, d/z \} \), and there is a migrated instance \( \{ R_2, \{ on/R \} \} \) in \( T^{R_1} \) satisfying \( \phi_i \supseteq \{ on/R \} \).

The second method of shadowing uses an **origins set** (OS) that is associated with each proposition to keep track of and propagate propositional dependencies in an assumption-based truth maintenance system SNeBR [3]. In the above example, OS of \( R_1 \) is \( \{ R_1 \} \) and OS of \( R_2 \) is \( \{ R_1, trans(on) \} \) since \( R_2 \) is derived from the two propositions. From the viewpoint of deductive learning, propositional dependencies represented by OS can be regarded as a type of experience. A subset-superset comparison between OSs of two rules can be used to shadow a redundant rule. For instance, \( R_1 \) can be regarded as more general than \( R_2 \) since OS of \( R_1 \) is a proper subset of the OS of \( R_2 \). A general shadowing principle using OS is described below.

**Shadowing Principle 2** Let \( R_1, R_2, \ldots, R_k \) be rules that are all applicable by a query \( q \) at some point during a deduction. Also let \( \Omega_1, \Omega_2, \ldots, \Omega_k \) be origin sets of each \( R_i, 1 \leq i \leq k \). Then, a deduction branch activated by a rule \( R_i \) is shadowed from the inference when (1) there is a rule \( R_j \) \( (j \neq i, 1 \leq j \leq k) \) such that \( \Omega_j \supseteq \Omega_i \), and (2) the outermost quantifier variables of \( R_i \) which also appear in \( R_j \) are bound by the pattern matching of \( q \) and a consequent of \( R_i \).

The third method of shadowing uses the concept of common instances between two patterns. Matching between two patterns \( S \) and \( T \) is a function \( \text{mgci} \) of two patterns \( S \) and \( T \) such that \( \text{mgci} \). If \( \text{mgci} = T \odot S \). For example, a matching between \( P(x,b) \) and \( P(a,y), \) where \( x \) and \( y \) are variables, produces \( \sigma = \{ a/x \} \) and \( \tau = \{ b/y \} \), and the mgci's of these two is \( P(a,b) \). We are striving to find a way of blocking deduction chains by using the mgci of two patterns, and obtain the following principle.

**Shadowing Principle 3** A deduction branch initiated by a rule \( R \) is shadowed when the mgci of the query \( q \) and a consequent of \( R \) is ground and already asserted in the knowledge base.

In the transitive rule example when \( R_1 \) and \( R_2 \) co-exist, the query \( R(a,c) \), where \( R \) is a variable, will shadow \( R_2 \) since the mgci of \( R(a,c) \) on \( z \) is \( on(a,c) \) which is a ground instance and already asserted.

For all the three shadowing principles, we claim that a deduction branch shadowed by any shadowing principle would never have produced any new results that cannot be produced by non-shadowed branches.

Implementation of knowledge migration and shadowing in SNePS [5] is in progress, and a preliminary result of performance improvement is shown in [1].

**References**


