Modeling a Theory of Second Language Acquisition in ASP

Daniela Inclezan
Computer Science Department
Texas Tech University
Lubbock, TX 79409 USA
daniela.inclezan@ttu.edu

Abstract

This paper is a contribution to the line of research opened by Balduccini and Girotto (2010; 2011), which explores the suitability of Answer Set Programming (ASP) for the formalization of scientific theories about the mind. We investigate the suitability of ASP in modeling one of the dominant theories about second language acquisition: Input Processing (VanPatten 2004). This theory is formulated as a series of default statements that rely on what is assumed to be the learners' knowledge about the world and the second language. We report on an application of our model to predicting how learners of English would interpret sentences containing the passive voice and present a system, $PIas$, that uses these predictions to assist language instructors in designing teaching materials.

Introduction

This paper is a contribution to the line of research opened by Balduccini and Girotto (2010; 2011), which explores the suitability of Answer Set Programming (ASP) (Gelfond and Lifschitz 1991; Niemelä 1998; Marek and Truszczynski 1999) for the formalization of scientific theories about the human mind. As pointed out in the cited articles, a substantial number of theories that are of a qualitative nature are formulated in natural language, often in the form of defaults. Modeling these theories in a precise mathematical language would allow scientists to accurately study and test their statements. ASP seems to be a suitable tool for this task, as it provides the means for an elegant and precise representation of defaults. Furthermore, there are known methodologies for knowledge representation using ASP that may be relevant for some theories.

In this work, we explore whether ASP alone is sufficient to model one of the dominant theories about second language acquisition—Input Processing (VanPatten 2004)—or if extensions of ASP are necessary for the task. In this context, the expression "second language" refers to any language that is learned after the first one. "Second language acquisition" is the process by which people learn a second language. Input Processing (IP) is a theory about how learners of a second language process input in that language. It describes the strategies that second language learners use to get meaning out of written or spoken text in the second language in on-line comprehension, given their limitations in vocabulary, working memory, or internalized knowledge of grammatical structures. As a result of applying these strategies, learners do not always come up with the correct interpretation of input sentences, and this is assumed to cause delays in the acquisition of certain grammatical structures.

For instance, IP predicts that beginner learners of English reading the sentence *"The cat was bitten by the dog"* would only be able to retrieve the meanings of the words *"cat"*, *"bitten"*, and *"dog"* and end up with something like the sequence of concepts CAT-BITE-DOG. Although they may notice the word *"was"* or the ending *"-en"* of the verb *"bitten"*, they would not be able to process them (i.e., connect them with the function they serve, which is to indicate passive voice) because of limitations in processing resources. Next, IP predicts that the sentence above, now mapped into the sequence of concepts CAT-BITE-DOG, would be incorrectly interpreted by these learners as *"The cat bit the dog"* because of a hypothesized strategy of assigning agent status to the first noun encountered in a sentence.

IP, as described in (VanPatten 2004), consists of two principles formulated as defaults. Each principle contains subprinciples that represent refinements of or exceptions to the original defaults. For example, a sub-principle of IP predicts that beginner learners of English would correctly interpret the sentence *"The shoe was bitten by the dog"* because agent status cannot be assigned to the first noun, as a shoe cannot bite. This can happen even if the learner has not yet internalized the structure of the passive voice in English or did not have the resources to process it in the above sentence. Similarly for the sentence *"The man was bitten by the dog"* because it is unlikely for a man to bite a dog. These strategies can also be applied to stories consisting of several sentences where information from previous sentences conditions the interpretation of later ones. For example, the second sentence of the story: *"The cat killed the dog. Then, the dog was bitten by the cat."* would be interpreted correctly even by beginner learners, because a dead dog cannot bite.

ASP seems to be a natural choice for modeling this theory, as defaults and their exceptions can be represented in ASP in an elegant and precise manner. Furthermore, IP assumes that learners possess some commonsense knowledge about actions and dynamic domains. For instance, they know un-

der what conditions a biting action can occur. In ASP, there is substantial research on how to represent actions and dynamic domains in which change is caused by actions. We will illustrate how we used these features of ASP to create a formalization of IP. Our formalization allowed us to notice some vague areas in the theory and to make predictions about how learners would interpret simple sentences and paragraphs containing the passive voice in English. We used these predictions to create a system, $PIas$, that can assist language teachers in designing instructional materials. $PIas$ relies on the guidelines of an established teaching method, Processing Instruction (PI) (VanPatten 1993), that is based on the principles of Input Processing. This method says that a sentence is valuable for teaching and testing if learners cannot interpret it correctly unless they have internalized the appropriate grammatical structures.

In what follows, we give an introduction to ASP, describe our encoding of the learners' background knowledge, and present our formalization of the IP principles. We test our model by comparing its predictions with the ones of IP. We present the system $PIas$ and end with discussions and directions for future work.

Answer Set Programming

Answer Set Programming (ASP) (Niemelä 1998; Marek and Truszczynski 1999) is a declarative programming paradigm based on the answer set semantics (Gelfond and Lifschitz 1991). Before we define programs of ASP, we introduce some preliminary definitions. A signature is a collection of constant, function, predicate, and variable symbols. Terms and atoms over this signature are defined as in predicate logic. A literal is an atom a or its negation $\neg a$, where the symbol $\neg$ denotes strong negation. A rule over a signature Σ is a statement of the form:

$$l_1 \vee \ldots \vee l_i \leftarrow l_{i+1}, \ldots, l_m, \text{not } l_{m+1}, \ldots, \text{not } l_n$$

where l_1, ..., l_n are literals of Σ and the symbol "not " denotes default negation.

A *program* of ASP is a pair $\langle \Sigma, \Pi \rangle$, where Σ is a signature and Π is a collection of rules over Σ.

For the formal semantics of programs of ASP we refer the reader to (Gelfond and Lifschitz 1991). For brevity, we only include here the informal semantics. A program can be seen as a specification for the set(s) of beliefs held by a rational agent. An *answer set* A of a logic program $\langle \Sigma, \Pi \rangle$ is a consistent set of literals of Σ that corresponds to a set of beliefs held by the rational agent, such that A satisfies the rules of Π and the rationality principle: *"Believe nothing you are not forced to believe."*

The properties of ASP programs and the methodology of representing knowledge in ASP were the subject of extensive research. ASP is particularly suitable for default reasoning, reasoning with incomplete or non-deterministic information, and representing and reasoning about dynamic domains. Another advantage of ASP is that different reasoning tasks can be reduced to computing answer sets of a logic program, which can be done using general purpose ASP solvers.

Logic Form of Input

The input for our model of the IP theory is a logic form encoding of sentences and paragraphs. Our logic form contains two sorts, $sentence$ and $paragraph$. The sentence structure is represented using the predicate $word$, where $word(K, S, W)$ means that the K^{th} word of sentence S is W; we assume that the index K starts at 1. For example, the sentence *"The cat was bitten by the dog"* in the introduction is encoded as:

$sentence(s)$.
$word(1, s, \text{“}the\text{”})$. $word(2, s, \text{“}cat\text{”})$.
$word(3, s, \text{“}was\text{”})$. $word(4, s, \text{“}bitten\text{”})$.
$word(5, s, \text{“}by\text{”})$. $word(6, s, \text{“}the\text{”})$. $word(7, s, \text{“}dog\text{”})$.

The structure of a paragraph is encoded using the predicate $sent(K, P, S)$ – the K^{th} sentence of paragraph P is S. A paragraph p with two sentences, s_1 and s_2, is defined as:

$paragraph(p)$.
$sent(1, p, s_1)$. $sent(2, p, s_2)$.

For any input sentence or paragraph X, by $lp(X)$ we denote the logic form encoding of X.

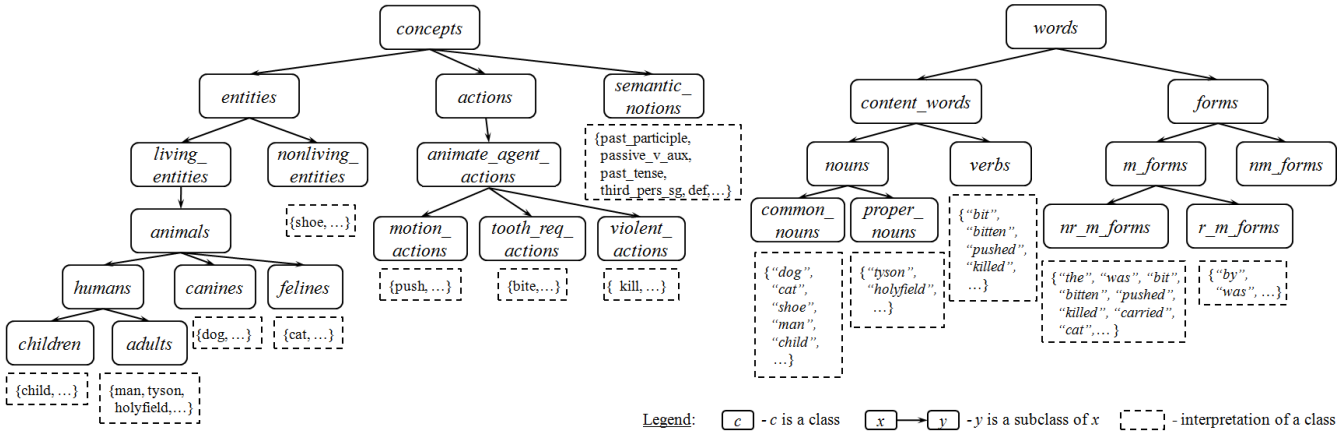
Learners' Background Knowledge

The IP theory assumes that, in their minds, the learners possess some background knowledge about the world and its dynamics, and a linguistic system of the second language containing, among others, a complex mental dictionary. As it is not clear how this knowledge is actually encoded in the mind, we represent it using known methodologies from the field of nonmonotonic reasoning, and, in particular, from the area of knowledge representation using ASP. We are not trying to replicate the human mind, but rather create a representation that would produce the same results in relation to the IP theory as the ones demonstrated by empirical studies and illustrated by the sentences presented in the introduction.

Inheritance Hierarchies

We assume that a learner's knowledge base contains classes of concepts (i.e., cognitive constructs) organized into a DAG-like inheritance hierarchy C with root $concepts$ with three children: $actions$, $entities$, and $semantic\_notions$. Instances of the latter class are possible meanings of grammatical structures (e.g., the concept of "some other person than the speaker and the addressee", referred to as third-person singular). Classes may have subclasses, as you can see in Fig. (1)(a). This is not a complete picture of C but rather a sufficiently detailed one for the later examples.

Additionally, the learner's knowledge base contains classes of words of a language, organized into a tree-like inheritance hierarchy L with root $words$, with two children: $content\_words$ and $forms$. Content words are those that carry the meaning of a sentence: nouns, verbs, adjectives, and adverbs. Forms, also called "grammatical structures", are inflections, articles, or particles. For example, the third-person-singular marker *"-s"* attached to verbs as in *"makes"* and the article *"the"* are forms. Based on the formulation of IP, we assume that learners that have acquired



(a) Instance of Hierarchy C (b) Instance of Hierarchy L

Figure 1: Instances of Hierarchies C and L

a form also know implicitly whether that form is meaningful, which means that it contributes meaning to the overall comprehension of a sentence, or not. Similarly, we assume that they are able to distinguish between redundant and nonredundant forms, where a redundant form is one whose meaning can usually be retrieved from other parts of a sentence. A sufficiently detailed view of L for our examples appears in Fig.(1)(b), where m_forms stands for meaningful forms, nm_forms for nonmeaningful, nr_m_forms for nonredundant meaningful, and r_m_forms for redundant meaningful forms. In what follows, we will use the term *word categories* (or just *categories*) for L's classes.

We model the two inheritance hierarchies in ASP by applying common practices (Baral 2003). We denote the resulting logic program by $\mathcal{H}$. The signature of $\mathcal{H}$ contains class names from the two hierarchies; a sort $class$; a relation $is\_subclass(C_1, C_2)$ (C_1 is a subclass of C_2); and its transitive closure $subclass(C_1, C_2)$. $\mathcal{H}$ contains rules for $subclass$ and, for each hierarchy, facts of the type:

$$is\_subclass(entities, concepts).$$

Next, we consider *instances of hierarchies* C and L. Informally, an instance of a hierarchy is an interpretation of its classes, given a universe, such that it respects the $subclass$ relation and the interpretation of the root class is equal to the universe. In addition, although L is a tree, we assume that the interpretations of its leaves are not disjoint. The reason behind this assumption is that, instead of separating a word into its smaller parts (e.g., *"bitten"= "bite"+"-en"*), we say that an inflected word belongs to two classes (e.g., we say that *"bitten"* is both a $verb$ and a nr_m_form, which means a nonredundant meaningful form). We do so for simplicity: word formation rules are not trivial and representing them is outside the scope of this paper. Note that, for a person, L may have multiple instantiations: one for each language he knows. Given that our examples will be about passive voice in English, we only consider here L's instantiation for English that a learner may possess.

To model instances of the two hierarchies, we extend the signature of $\mathcal{H}$ by a sort $object$, a relation $is\_a(X, C)$ (object X is a C) denoting links between objects and leaf classes, and $inst(X, C)$ (X is an instance of C), for the closure of is_a. We add to $\mathcal{H}$ rules for $inst$ and facts of the form:

$$is\_a(dog, canines).\ \ is\_a(\text{“}dog\text{”}, common\_nouns).$$

Note that we use constants (e.g., dog) for concepts and string constants (e.g., “dog”) for words. Here, an instance dog refers to a prototypical dog. The instantiation of L will differ for learners of different levels of proficiency: beginner learners will know less content words than advanced learners and very few forms. The instance in Fig.(1)(b) corresponds to an advanced learner.

Finally, we need to model the connections learners make between words and concepts. For simplicity, we assume that all the connections in a learner's knowledge base are correct. As the same word can belong to different leaves of hierarchy L, in our formalization we do not associate words to concepts, but instead specify associations of concepts to *pairs* formed by a word and a leaf word category. Informally, if w is a word, ctg is a leaf word category, and the pair $\langle w, ctg\rangle$ is connected to a concept c, we say that w interpreted as a member of ctg has the meaning c. For example, we say that “$bitten$” interpreted as a content word has the meaning $bite$; the same word interpreted as a nonredundant meaningful form has the meaning $past\_participle$. To represent these associations, we extend the signature of $\mathcal{H}$ by a relation $has\_meaning(W, Ctg, C)$ saying that word W interpreted as a member of category Ctg has the meaning C. We require Ctg to be a leaf of L. $\mathcal{H}$ contains facts like:

$$has\_meaning(\text{“}bitten\text{”}, verbs, bite).$$
$$has\_meaning(\text{“}bitten\text{”}, nr\_m\_forms, past\_participle).$$

We add a relation $meaning(W, Ctg, C)$ to extend $has\_meaning$ to super-classes of the word category Ctg. Nouns are associated with entities, verbs with actions, and forms with semantic notions. The program $\mathcal{H}$ contains more such associations for advanced learners than for beginners.

Action Descriptions

The learners' background knowledge contains, besides the two instantiated hierarchies mentioned above, some representation of action preconditions and effects. As shown in the introduction, learners use this knowledge in processing isolated sentences and stories. *We formalize this knowledge using standard methodologies from the field of representing and reasoning about action and change, more specifically from the area of representing knowledge about discrete dynamic systems in ASP.*

Discrete dynamic domains are domains in which change is caused by actions in a way that can be modeled by transition diagrams whose nodes are states of the world and whose arcs are labeled by actions. As transition diagrams can easily become unmanageable in size, such domains are normally described using an *action description* – a collection of axioms of some action language (Gelfond and Lifschitz 1998). The signature of an action description specifies the objects of the domain, their relevant properties, and possible actions of the domain. There are two types of properties: *statics*, the value of which cannot be changed, and *fluents*, which can be changed by actions. Axioms of an action description specify the direct and indirect effects of actions, and action preconditions. They are eventually translated into logic programming rules. In this paper, we consider a translation into ASP similar to the one in (Balduccini and Gelfond 2003) and only show the translated rules due to space constraints.

In our case, objects of the domain are instances of classes $entities$ and $actions$ in hierarchy C. We will refer to actions of the domain as *events* in order to be consistent with the terminology used in the IP theory and to distinguish them from instances of the class $actions$. Events of the domain are terms of the type $ev(A, E_1, E_2)$ where A is an instance of class $actions$ denoting a type of action, and E_1 and E_2 are instances of the class $entities$ denoting the agent and the object of that action, respectively. Informally, the term $ev(bite, cat, dog)$ denotes the event of "the cat biting the dog". The term $ev(bite, shoe, dog)$ denotes the event of "the shoe biting the dog", which cannot occur in real life, but is an event that people can describe in a sentence. We only consider events that are denoted by transitive verbs (i.e., requiring an agent and an object) because this is the type of events that appears in the examples of the IP theory. Properties of the domain are also denoted by terms. Some examples are the fluents: $alive(x)$, $can\_move(x)$ (x is able to move by itself), and the static $has\_teeth(x)$.

Our action description, called $\mathcal{A}$, is a logic program whose signature contains: the above mentioned objects, events, and properties; a sort called $step$; a relation $holds(F, I)$ (fluent F holds at step I); and a relation $occurs(Ev, I)$ (event Ev occurs at step I of the story). $\mathcal{A}$ contains rules describing the *direct effects of actions*, which have the form:

$$\neg holds(alive(E), I+1) \leftarrow occurs(ev(kill, E_1, E), I).$$

The informal reading is that killing an entity causes it to stop being alive. *Indirect effects of actions* are described in $\mathcal{A}$ via rules of the type:

$$\neg holds(can\_move(E), I) \leftarrow \neg holds(alive(E), I).$$

This says that a dead entity cannot move by itself. Finally, $\mathcal{A}$ contains rules describing *event preconditions*. To ease the later encoding of the IP theory, we introduce a separate relation $impossible$ and define $\neg occurs$ in terms of it:

$$\neg occurs(Ev, I) \leftarrow impossible(Ev, I).$$

$$impossible(ev(A, E_1, E_2), I) \leftarrow inst(A, animate\_agent\_actions), \neg inst(E_1, living\_entities).$$

$$impossible(ev(A, E_1, E_2), I) \leftarrow inst(A, animate\_agent\_actions), \neg holds(alive(E_1), I).$$

$$impossible(ev(A, E_1, E_2), I) \leftarrow inst(A, tooth\_req\_actions), \neg has\_teeth(E_1).$$

The second and third rules above say that the agent of an animate-agent action must in fact be a living entity that is alive when the event occurs. The last rule says that a tooth-requiring action cannot be performed if the agent has no teeth.

$\mathcal{A}$ contains the standard ASP encoding of the Inertia Axiom (Hayes and McCarthy 1969) for fluents, which consists of two defaults. We will later use the axioms of $\mathcal{A}$ given here to illustrate an application of our model of IP.

Learners also possess some knowledge about events that actually happened in the real world. We store this type of information in a logic program $\mathcal{G}$ as facts of the type:

$$hpd(ev(bite, tyson, holyfield)).$$ [1]

Informally, the fact above says that at some point in the past it happened that Mike Tyson bit Evander Holyfield.

$\mathcal{G}$ contains some additional information about what values a learner expects fluents to have at the beginning of a story encountered for the first time: unless otherwise specified in the text, he will believe that animals can move, living entities are actually alive, etc. Such defaults will be encoded as:

$$holds(can\_move(E), 0) \leftarrow inst(E, animals), \text{not } \neg holds(can\_move(E), 0).$$

where 0 is the time step of the first sentence. We separate $\mathcal{G}$ from $\mathcal{A}$ because, unlike $\mathcal{A}$, $\mathcal{G}$ does not specifies the transition diagram; it describes prior knowledge and preferred paths.

Defaults and Inheritance Hierarchies with Defaults

Finally, the IP theory assumes that learners possess some knowledge about what events are improbable to happen. We describe it in ASP using a known methodology for representing defaults (Baral and Gelfond 1994). We base our choice on two reasons. First, encoding this kind of knowledge using defaults is very close, in our opinion, to how humans think about probable and improbable events. Second, in the description and exemplification of the IP theory, the given explanations are of the type *"Normally, people do not bite animals"*, which are default statements. For example, we encode the default above as follows:

[1] In contrast with $hpd(Ev)$, the relation $occurs(Ev, I)$ is used with events extracted by a learner from a text. Such events are "hypothetical" in the sense that they did not necessarily happen in reality. This can be either because the text is not truthful or because the learner did not extract the correct meaning out of the text.

$$\neg occurs(ev(bite, E_1, E_2), I) \leftarrow$$
$$inst(E_1, humans),$$
$$inst(E_2, animals),$$
$$\text{not } occurs(ev(bite, E_1, E_2), I),$$
$$\text{not } ab(d_1(ev(bite, E_1, E_2), I)).$$

Next, we represent exceptions to these defaults. We can say that Mike Tyson is an exception, as he is known to have bitten other people (but not children). It is also known that children may bite other animals, especially at the toddler stage. These are *weak exceptions*, as they make the default in the rule above inapplicable. They are encoded as:

$$ab(d_1(ev(bite, tyson, E_2), I)) \leftarrow$$
$$inst(E_2, humans),$$
$$\neg inst(E_2, children).$$
$$ab(d_1(ev(bite, E_1, E_2), I)) \leftarrow$$
$$inst(E_1, children),$$
$$\text{not } ab(d_3(ev(bite, E_1, E_2), I)).$$

The class $children$ is a son of the class $humans$ in the hierarchy C. The last axiom, which is also a default, describes a case in which a property of the class $humans$ does not propagate down in the inheritance hierarchy.

The examples above illustrate how we applied known ASP techniques to represent defaults within an inheritance hierarchy, relevant for our formalization of IP.

The First Principle of Input Processing

Principle 1 of IP describes what chances words in a sentence have to get processed during online comprehension. It takes as an input a sentence and the background knowledge of a learner and produces a possibly partial mapping of words (interpreted according to some category) into concepts. This principle is listed in (VanPatten 2004) as follows:

1. The Primacy of Meaning Principle. Learners process input for meaning before they process it for form.

1a. The Primacy of Content Words Principle. Learners process content words in the input before anything else.

1b. The Lexical Preference Principle. Learners will tend to rely on lexical items[2] *as opposed to grammatical form to get meaning when both encode the same semantic information.*

1c. The Preference for Nonredundancy Principle. Learners are more likely to process nonredundant meaningful grammatical forms before they process redundant meaningful forms.

1d. The Meaning-Before-Nonmeaning Principle. Learners are more likely to process meaningful grammatical forms before nonmeaningful forms irrespective of redundancy.

1e. The Availability of Resources Principle. For learners to process either redundant meaningful grammatical forms or nonmeaningful forms, the processing of overall sentential meaning must not drain available processing resources.[3]

1f. The Sentence Location Principle. Learners tend to process items in sentence initial position before those in final position and these latter in turn before those in medial position (all other processing issues being equal).

Example 1. Let us show what predictions Principle 1 makes about the processing of words from the following sentence:

S_1. *The cat was bitten by the dog.*

According to 1a, content words (i.e., nouns, verbs, adjectives, and adverbs) have the highest chance of getting processed, in this case: *"cat"*, *"bitten"*, and *"dog"*. Among them, based on 1f, *"cat"* has the highest chance as it is in sentence initial position, followed by *"dog"* in final position, and then by *"bitten"* in medial position.

The next chance belongs to meaningful forms based on 1d, in this case: *"the"*, *"was"*, *"by"*, together with *"cat"* as an indicator of third-person singular, and *"bitten"* as an indicator of passive voice. According to 1c, out of these forms, the non-redundant ones are more likely to get processed, in particular the definite article *"the"* and the word *"cat"* as an indicator of third-person singular, both in initial position, followed by *"the"* in final sentence position, and then by the forms *"was"* as an indicator of past tense and *"bitten"* as an indicator of passive voice in medial position.

Principle 1e says that the whole sentence has the next chance of getting processed, followed by the redundant form *"by"*. Finally, according to 1b, the redundant form *"was"* as an indicator of third-person singular may or may not get processed, independently of available resources, because its meaning was already obtained from the word *"cat"*. Note that how many words, in their order or likelihood, actually get processed depends on the resource capacity of a learner.

Principle 1 and its corollaries specify a partial order between words in a sentence, given the category they belong to and their sentential position. Greater elements in this ordering have more chances of being processed than lesser elements.

We start formalizing Principle 1 by looking at its subprinciples 1a, 1c, and 1d. Together, they describe a partial order on word categories. To model it, we define a relation is_ml_ctg on categories, where $is\_ml\_ctg(Ctg_1, Ctg_2)$ says that words from category Ctg_1 are more likely to get processed than words from category Ctg_2. Based on Principle 1a, we have the fact:

$$is\_ml\_ctg(content\_words, forms).$$

From Principle 1d, we have that:

$$is\_ml\_ctg(m\_forms, nm\_forms).$$

and from Principle 1c:

$$is\_ml\_ctg(nr\_m\_forms, r\_m\_forms).$$

Next, we look at Principle 1f, which describes a similar partial order on sentence positions. To specify the different possible sentence positions, we define a sort $sentence\_position$ with three elements: $initial$, $medial$, and $final$. We use a relation $is\_ml\_pos(Pos_1, Pos_2)$, which says that words in sentence position Pos_1 are more likely to be processed than words in Pos_2 (as long as they belong to the same word category). We encode Principle 1f via the facts:

$$is\_ml\_pos(initial, final).$$
$$is\_ml\_pos(final, medial).$$

By ml_ctg and ml_pos, respectively, we denote the transitive closures of the two relations above. In addition, we ex-

[2] A *lexical item* is a word (e.g., *"cat"*) or sequence of words (e.g., *"look for"*) that is the basic unit of the mental vocabulary.

[3] *Processing resources* refers to resources available in the learner's working memory for processing on-line input.

tend the relation ml_ctg down to subclasses of categories, but not upwards to superclasses.

Based on the two relations above, we can now define the partial relation between words, given their category and position. The IP theory does not precisely say how many words starting from the beginning of a sentence are part of the initial sentence position. We define it as the first n words of a sentence, where n is a parameter of the encoding. Similarly for final positions. We use the relation $pos(K, S, Pos)$ to say that the K^{th} word of sentence S is in Pos sentence position. We introduce a relation $ml\_wrd(K_1, S, Ctg_1, K_2, Ctg_2)$, which says that the K_1^{th} word of S is more likely to get processed for its interpretation as an element of the category Ctg_1 than the K_2^{th} word of the same sentence for category Ctg_2. To simplify the encoding of future axioms, we limit this relation to categories $content\_words$, nr_m_forms, r_m_forms, and nm_forms, mentioned in Principle 1, which we consider to be of a special sort ctg_p1:

$$\begin{aligned} ml\_wrd(K_1, S, Ctg_1, K_2, Ctg_2) \leftarrow \\ & ctg\_p1(Ctg_1),\ \ ctg\_p1(Ctg_2), \\ & word(K_1, S, W_1),\ \ inst(W_1, Ctg_1), \\ & word(K_2, S, W_2),\ \ inst(W_2, Ctg_2), \\ & ml\_ctg(Ctg_1, Ctg_2). \end{aligned}$$

$$\begin{aligned} ml\_wrd(K_1, S, Ctg, K_2, Ctg) \leftarrow \\ & ctg\_p1(Ctg), \\ & word(K_1, S, W_1),\ \ inst(W_1, Ctg), \\ & word(K_2, S, W_2),\ \ inst(W_2, Ctg), \\ & pos(K_1, S, Pos_1),\ \ pos(K_2, S, Pos_2), \\ & ml\_pos(Pos_1, Pos_2). \end{aligned}$$

The first rule relates to Principles 1a, 1c, and 1d, as it is based on the ordering of categories; the second rule is about Principle 1f, as it uses the sentence position ordering, for a given category of words. The effects of the ordering ml_wrd on the processing of a sentence will be seen later, when we present a relation called $resources\_consumed$.

Next, we describe the conditions for a word to get processed (i.e., to be mapped into a concept). We introduce a relation $map(K, S, Ctg, C)$, which says that the K^{th} word of S was processed for its interpretation according to category Ctg and was mapped into concept C. We encode Principle 1 as:

$$\begin{aligned} map(K, S, Ctg, C) \leftarrow \\ & ctg\_p1(Ctg), word(K, S, W), inst(W, Ctg), \\ & meaning(W, Ctg, C), \\ & enough\_resources\_available(K, S, Ctg), \\ & \text{not } ab(d_{map}(K, S, Ctg, C)). \end{aligned}$$

This says that, normally, a word will be processed if there are enough resources available. Our representation of processing resources does not attempt to be a model of working memory as the IP theory focuses mainly on what the end result of processing a sentence in working memory is expected to be: certain word-to-concept associations will be made while others will not. We model processing resources using the unary predicate $resource\_capacity(N)$ (the maximum number of resources that a learner can use in processing one sentence is N) and making the simplifying assumption that the processing of any word or of the whole sentence takes up one resource. We add a predicate $resources\_consumed(N, K, S, Ctg)$, which says that N resources are consumed in processing more likely words than the K^{th} word of sentence S for category Ctg. The definition of this relation captures the implications of the ordering ml_wrd on the processing of words, as follows:

$$\begin{aligned} resources\_consumed(N, K, S, Ctg) \leftarrow \\ & ctg\_p1(Ctg),\ \ word(K, S, W),\ \ inst(W, Ctg), \\ & N = \#count\ \{ml\_wrd(K_1, S, Ctg_1, K, Ctg) : \\ & \qquad ctg\_p1(Ctg1)\ \}. \end{aligned}$$

($\#count$ is a function returning the cardinality of a set). We add a predicate, $resources\_\ required(N, K, S, Ctg)$ (N resources are necessary to process words that are equally likely to be processed as the K^{th} word of S for category Ctg). Then, $enough\_resources\_\ available(K, S, Ctg)$ is true if the capacity exceeds or is equal to the sum of resources consumed and required.

To model Principle 1e, we also count resources consumed on processing the whole sentence as part of our calculation for $enough\_resources\_available$:

$$\begin{aligned} enough\_resources\_available(K, S, Ctg) \leftarrow \\ & resources\_consumed(N_1, K, S, Ctg), \\ & resources\_consumed\_sent(N_2, S, Ctg), \\ & resources\_required(N_3, K, S, Ctg), \\ & resource\_capacity(N), N - N_1 - N_2 \geq N_3. \end{aligned}$$

and define these consumed resources to be equal to 1 for categories r_m_forms (i.e., redundant meaningful forms) and nm_forms (i.e., nonmeaningful forms), and 0 otherwise.

The only remaining principle is 1b. Based on its accompanying explanation in (VanPatten 2004), its meaning is that a form that is *normally* redundant may not be processed at all if it is *actually* redundant in that sentence (i.e., its meaning was already extracted from some other word). We encode this knowledge as a *possible* weak exception to the default in the rule about map via a disjunctive rule:

$$\begin{aligned} ab(d_{map}(K, S, Ctg, C)) \vee \neg ab(d_{map}(K, S, Ctg, C)) \leftarrow \\ & word(K, S, W),\ \ meaning(W, Ctg, C), \\ & word(K_1, S, W_1),\ \ K \neq K_1,\ \ W \neq W_1, \\ & map(K_1, S, Ctg_1, C). \end{aligned}$$

The informal reading of this axiom is that meanings that are actually redundant in a sentence may or may not be exceptions to the default for relation map.

This ends our formalization of Principle 1 and its corollaries. We collect the rules above in a program $\mathcal{P}_1$.

The Second Principle of Input Processing

Principle 2 describes the strategies that learners use to extract the meaning of a sentence, given the words they were able to process. The input of Principle 2 is the output of Principle 1 for a given sentence (i.e., a mapping of words to concepts). Its output is an event denoting the meaning extracted by the learner from the sentence. This principle is listed in (VanPatten 2002; 2004) as follows:

2. The First Noun Principle (FNP). Learners tend to process the first noun or pronoun they encounter in a sentence as the agent.

2a. The Lexical Semantics Principle. Learners may rely on lexical semantics,[4] *where possible, instead of on word order to interpret sentences.*

2b. The Event Probabilities Principle. Learners may rely on event probabilities, where possible, instead of on word order to interpret sentences.

2c. The Contextual Constraint Principle. Learners may rely less on the First Noun Principle if preceding context constrains the possible interpretation of a clause or sentence.

2d. Prior Knowledge. Learners may rely on prior knowledge, where possible, to interpret sentences.

2e. Grammatical Cues. Learners will adopt other processing strategies for grammatical role assignment only after their developing system[5] *has incorporated other cues.*

Example 2. We illustrate the predictions made by Principle 2 for several sentences. First, we consider the case of beginner learners, who have limited resources and vocabulary, and can only process the content words out of a sentence. Based on Principle 1, beginners would map the words *"cat"*, *"bitten"*, and *"dog"* in sentence S_1 from Example 1 into the concepts cat, $bite$, and dog respectively and would not be able to process any other words. Principle 2 predicts that beginners would assign agent status to the first noun in S_1 and hence interpret S_1 incorrectly as *"The cat bit the dog."*

Beginners are expected to correctly interpret the sentence:

S_2. *The shoe was bitten by the dog.*

as a shoe cannot bite a dog (lexical semantics). According to Principle 2a, lexical semantics overrides the assignment of agent status to the first noun. The sentence:

S_3. *The man was bitten by the dog.*

is also supposed to be interpreted correctly by beginners because men normally do not bite animals (event probabilities and Principle 2b). Similarly for the sentence:

S_4. *The man was bitten by the child.*

where we see the application of defaults in an inheritance hierarchy. Principle 2d predicts the correct interpretation of:

S_5. *Holyfield was bitten by Tyson.*

assuming that learners have the prior knowledge that Tyson bit Holyfield.

Let us now consider some short paragraphs:

P_1. (S_6.) *The cat pushed the dog.* (S_7.) *Then, the dog was bitten by the cat.*

Sentence S_7 is supposed to be incorrectly interpreted by beginners because none of the Principles 2a-e applies. Instead, the second sentence of the paragraph:

P_2. (S_8.) *The cat killed the dog.* (S_9.) *Then, the dog was bitten by the cat.*

would be interpreted correctly due to lexical semantics in context, as predicted by Principles 2a and 2c together.

[4] *Lexical semantics* refers to the meaning of lexical items.

[5] *Developing system* refers to the representation of grammatical knowledge in the mind of the second language learner. This representation changes as the learner acquires more knowledge.

Finally, let us consider advanced learners who possess enough resources and a large vocabulary, which allow them to map all words in a sentence into concepts. According to Principle 2e, these learners are expected to interpret all above sentences correctly, as they are able to detect the use of the active or passive voice and can rely on grammatical cues for sentence interpretation.

Principle 2 is a default statement and its sub-principles express exceptions to it. Our encoding will be based on some auxiliary relations: (1) $first\_noun(K, S)$ (the K^{th} word of sentence S is its first noun), and similarly for $verb(K, S)$, $second\_noun(K, S)$; and (2) a relation $contains\_concepts(S, E_1, A, E_2)$ (sentence S contains entity concept E_1 expressed by the first noun, action concept A expressed by its verb, and entity concept E_2 by its second noun).

The meaning of a sentence is an event. We remind the reader that we limited ourselves to events formed by actions requiring an agent and an object, and that $ev(A, E_1, E_2)$ denotes the event of action A being executed by agent E_1 onto object E_2. In what follows, if a sentence S contains concepts E_1, A, and E_2, in this order, we will call $ev(A, E_1, E_2)$ the *direct meaning* of S and $ev(A, E_2, E_1)$ its *reverse meaning*.

To encode Principle 2, we use a relation $extr\_m(Ev, S, fnp)$ saying that the learner extracted the meaning Ev from sentence S by applying the First Noun Principle (FNP):

$$extr\_m(ev(A, E_1, E_2), S, fnp) \leftarrow contains\_concepts(S, E_1, A, E_2), \text{not } extr\_m(ev(A, E_2, E_1), S, fnp).$$

The rule says that learners applying the FNP will extract the direct meaning from a sentence, unless they extract the reverse meaning. The extraction of the reverse meaning is encoded by several rules presented below.

We represent Principle 2a using the axiom:

$$extr\_m(ev(A, E_2, E_1), S, fnp) \leftarrow contains\_concepts(S, E_1, A, E_2), impossible(ev(A, E_1, E_2), I), \text{not } impossible(ev(A, E_2, E_1), I).$$

Informally, it says that learners will assign the reverse meaning to a sentence if this is a possible meaning and the direct meaning is impossible.

The formalization of Principle 2b will be similar:

$$extr\_m(ev(A, E_2, E_1), S, fnp) \leftarrow contains\_concepts(S, E_1, A, E_2), \text{not } impossible(ev(A, E_1, E_2), I), \neg occurs(ev(A, E_1, E_2), I), \text{not } hpd(ev(A, E_1, E_2)), \text{not } \neg occurs(ev(A, E_2, E_1), I).$$

I.e., a sentence will be assigned its reverse meaning if the direct meaning is possible, unlikely, and not known to have actually happened, while the reverse meaning may hypothetically occur (i.e., it is possible and not unlikely).

Principle 2d is encoded as follows:

$$extr\_m(ev(A, E_2, E_1), S, fnp) \leftarrow contains\_concepts(S, E_1, A, E_2), \text{not } impossible(ev(A, E_1, E_2), I), \text{not } impossible(ev(A, E_2, E_1), I), hpd(ev(A, E_2, E_1)).$$

This says that a learner using the FNP will extract the reverse meaning if he knows that this event actually happened.

The preference for grammatical cues when such cues can be interpreted (Principle 2e) is encoded via the rules:

$$\begin{array}{lcl} extr\_m(Ev, S) & \leftarrow & extr\_m(Ev, S, grm\_cues). \\ extr\_m(Ev, S) & \leftarrow & extr\_m(Ev, S, fnp), \\ & & \text{not } extr\_m\_by(S, grm\_cues). \\ extr\_m\_by(S, X) & \leftarrow & extr\_m(Ev, S, X). \end{array}$$

where $extr\_m(Ev, S)$ says that Ev is the meaning extracted from S; $extr\_m(Ev, S,\ grm\_cues)$ – the meaning Ev was extracted from S based on grammatical cues; and $extr\_m\_by(S, X)$ – the meaning of S was extracted based on strategy X. We exemplify the rules defining $extr\_m(Ev,$ $S, grm\_cues)$ for the case of the English passive voice:

$$\begin{array}{l} extr\_m(ev(A, E_1, E_2), S, grm\_cues) \leftarrow \\ \qquad contains\_concepts(S, E_1, A, E_2), \\ \qquad voice(S, active). \\ extr\_m(ev(A, E_2, E_1), S, grm\_cues) \leftarrow \\ \qquad contains\_concepts(S, E_1, A, E_2), \\ \qquad voice(S, passive). \end{array}$$

For this particular definition we used the relation $voice(S,$ $passive/active)$ (S is formulated in passive/active voice). We say that a learner detected the passive voice in a sentence if he was able to process the past participle (e.g., *"bitten"*) and passive voice auxiliary (e.g., *"was"*) appearing in it. He detected the active voice if he was able to process forms and he did not encounter the two markers for passive voice.

In our formalization of FNP, Principle 2c was embedded in the representation of Principles 2a, 2b, and 2d. The one thing left for contextual constraints is to record the events corresponding to the meaning extracted from previous sentences of the story, assuming the story time starts at step 0:

$$\begin{array}{lcl} occurs(Ev, I-1) & \leftarrow & extr\_m(Ev, S), \\ & & paragraph(P), \\ & & sent(I, S, P). \end{array}$$

The rules for Principle 2 are collected in a logic program $\mathcal{P}_2$.

Predictions about the English Passive Voice

We used our model of the IP theory to generate automated predictions about how sentences like the ones in Examples 1 and 2 would be interpreted by learners of English. We tested our formalization of IP by verifying that our automated predictions matched with what the theory predicts. We considered two different versions of $\mathcal{H}$: one for advanced learners ($\mathcal{H}_{adv}$) and another one for beginners ($\mathcal{H}_{beg}$). $\mathcal{H}_{adv}$ contains all the instances shown in Fig.1; $\mathcal{H}_{beg}$ contains only instances of content words. For each type of learner, we created a logic program Π (indexed by either adv or beg) by putting together: the corresponding encoding $\mathcal{H}$ of hierarchies C and L with their instances; the action description $\mathcal{A}$; the encoding of defaults $\mathcal{D}$; the prior knowledge $\mathcal{G}$; and the formalizations of the two principles $\mathcal{P}_1$ and $\mathcal{P}_2$.

For any input sentence or paragraph X, the answer set(s) of the program $\Pi \cup lp(X)$ corresponds to predictions of the IP theory about how a learner would interpret X.

We first run tests for Principle 1 by using sentence S_1 from Example 1, copied here with its words annotated by their sentential indices for a better understanding of the results: S_1. *The*$_1$ *cat*$_2$ *was*$_3$ *bitten*$_4$ *by*$_5$ *the*$_6$ *dog*$_7$. We considered different resource capacities and set the sentence position parameter n to 2. The answer sets of $\Pi_{adv} \cup lp(S_1)$ contained the map facts presented in Table 1, where an atom $map(k, s_1, ctg, c)$ in the answer set for capacity m indicates that the k^{th} word of S_1 will get processed by a learner with capacity m. Note that the map atom on the last line of the table is marked with an asterisk because two answer sets are generated for this capacity and this atom is part of one of them but not the other. Next, we tested our predictions for Principle 2 for beginners and advanced learners. The relevant parts of the answer sets for the sentences and paragraphs in Example 2 can be seen in Table 2. Our automated predictions matched the ones in Examples 1 and 2, which suggests that our model of IP is correct.

Table 1: Automated Predictions for Principle 1

Capacity	**Additional map Facts in the Answer Set(s) w.r.t. the Answer Sets for Smaller Capacities**
0	$\emptyset$
1	$map(2, s_1, content\_words, cat)$
2	$map(7, s_1, content\_words, dog)$
3	$map(4, s_1, content\_words, bite)$
9	$map(1, s_1, nr\_m\_forms, def)$ $map(2, s_1, nr\_m\_forms, third\_pers\_sg)$ $map(6, s_1, nr\_m\_forms, def)$ $map(3, s_1, nr\_m\_forms, passive\_v\_aux)$ $map(3, s_1, nr\_m\_forms, past\_tense)$ $map(4, s_1, nr\_m\_forms, past\_participle)$
11	$map(5, s_1, r\_m\_forms, agency)$ $map(3, s_1, r\_m\_forms, third\_pers\_sg)^*$

Table 2: Automated Predictions for Principle 2

X	**Answer Set of $\Pi_{beg} \cup lp(X)$ contains**	**Notes**
S_1	$extr\_m(ev(bite, cat, dog), s_1)$ $extr\_m\_by(s_1, fnp)$	Wrong Interp.
S_2	$extr\_m(ev(bite, dog, shoe), s_2)$ $impossible(ev(bite, shoe, dog), 0)$	Correct Interp.
S_3	$extr\_m(ev(bite, dog, man), s_3)$ $\neg occurs(ev(bite, man, dog), 0)$	Correct Interp.
S_4	$extr\_m(ev(bite, child, man), s_4)$	Correct
S_5	$extr\_m(ev(bite, tyson, holyfield), s_5)$ $hpd(ev(bite, tyson, holyfield))$	Correct Interp.
P_1	$extr\_m(ev(push, cat, dog), s_6)$ $extr\_m(ev(bite, dog, cat), s_7)$	Wrong Interp.
P_2	$extr\_m(ev(kill, cat, dog), s_8)$ $extr\_m(ev(bite, cat, dog), s_9)$ $\neg holds(alive(dog), 1)$ $impossible(ev(bite, dog, cat), 1)$	Correct Interp.
X	**Answer Set of $\Pi_{adv} \cup lp(X)$ contains**	**Notes**
S_1	$extr\_m(ev(bite, dog, cat), s_1)$ $extr\_m\_by(s_1, grm\_cues)$	Correct Interp.

The System $PIas$

We created a system, $PIas$, designed to assist instructors in preparing materials for the passive voice in English. $PIas$ follows the guidelines of a successful teaching method called Processing Instruction (PI) (VanPatten 2002), developed based on the principles of IP. For a sentence to be *valuable* in this approach, it must lead to the incorrect interpretation when grammatical cues are not used and the FNP is. S_1 above is such an example; sentences S_2 to S_5 are not.

$PIas$ has two functions. The first one is to specify whether sentences and paragraphs created by instructors are valuable or not. *This is relevant because even instructors trained in PI happen to create bad materials.* We define:

$$\begin{aligned} valuable(S) \quad \leftarrow \quad & extr\_m(Ev_1, S, grm\_cues), \\ & extr\_m(Ev_2, S, fnp), \\ & Ev_1 \neq Ev_2,\ voice(S, passive). \end{aligned}$$

This says that a sentence in the passive voice is valuable if its interpretation using grammatical cues (the correct one) does not coincide with the one based on FNP. We create a module $\mathcal{M}$ containing the definition above and its extension to paragraphs. $PIas$ takes as an input a sentence or paragraph X in natural language, encodes it in its logic form $lp(X)$, and computes the answer sets of a program consisting of Π_{adv}, $\mathcal{M}$, and $lp(X)$. X is valuable if the atom $valuable(X)$ belongs to all the answer sets of the resulting program.

The second function of $PIas$ is to generate all valuable sentences given a vocabulary and some simple grammar. *This is important because PI requires to expose learners to a large number of valuable sentences.* We add to $\mathcal{M}$ rules for sentence creation. For instance, one particular type of sentence is generated by:

$$\begin{aligned} sentence(s(\text{“}The\text{”}, N_1, \text{“}was\text{”}, V, \text{“}by\text{”}, \text{“}the\text{”}, N_2)) \leftarrow \\ schema(N_1, V, N_2). \\ word(1, s(\text{“}The\text{”}, N_1, \text{“}was\text{”}, V, \text{“}by\text{”}, \text{“}the\text{”}, N_2), \text{“}the\text{”}) \\ \leftarrow schema(N_1, V, N_2). \qquad \ldots \end{aligned}$$

where $schema(N_1, V, N_2)$ is true if N_1 and N_2 are common nouns and V is a verb in the past participle form (e.g., *“bitten”*). Atoms of the type $valuable(S)$ in the answer set(s) of the program $\Pi_{adv} \cup \mathcal{M}$ give all the valuable sentences that can be generated using our grammar and vocabulary.

Discussions and Future Work

This paper has shown that the use of ASP and its established methodologies for knowledge representation allowed us to formalize an important theory about second language acquisition in a natural way, without the need for extensions of ASP. Predicting how learners would interpret sentences in the second language was reduced to the task of computing answer sets of a logic program. By formalizing the IP theory, we were able to notice some of its vague areas. E.g., the notion of “initial sentence position” is not precisely defined and the theory does not clearly specify what amount of resources learners allocate for noticing words, accessing their meaning, processing them, etc. The only hint indicates that the simple processing of content words drains the resources of beginner learners. In such cases, we refined the theory by providing specific values to these parameters. Tuning them in the future can contribute to the development of IP theory.

This work can be expanded in various ways. We plan to improve the system $PIas$ to generate even more complex instructional materials according to the Processing Instruction teaching method. We also intend to refine the formalization of resources. Our current model does not capture cases when a learner hears a sentence or parts of it twice and the initial processing carries over to the second instance. Finally, it would be interesting to explore how we could make use of the short and long-term memory models described in Balduccini and Girotto (2010; 2011). The potential connection we see comes from the fact that the IP theory uses the concept of working memory, which is based on that of short-term memory.

Acknowledgments The author warmly thanks Michael Gelfond and Marcello Balduccini for their valuable suggestions.

References

Balduccini, M., and Gelfond, M. 2003. Diagnostic reasoning with A-Prolog. *TPLP* 3(4–5):425–461.

Balduccini, M., and Girotto, S. 2010. Formalization of Psychological Knowledge in Answer Set Programming and its Application. *TPLP* 10(4–6):725–740.

Balduccini, M., and Girotto, S. 2011. ASP as a Cognitive Modeling Tool: Short-Term Memory and Long-Term Memory. In Balduccini, M., and Son, T. C., eds., *Logic Programming, Knowledge Representation, and Nonmonotonic Reasoning*, volume 6565 of *LNCS*, 377–397. Berlin: Springer.

Baral, C., and Gelfond, M. 1994. Logic Programming and Knowledge Representation. *Journal of Logic Programming* 19(20):73–148.

Baral, C. 2003. *Knowledge Representation, Reasoning, and Declarative Problem Solving*. Cambridge University Press.

Gelfond, M., and Lifschitz, V. 1991. Classical Negation in Logic Programs and Disjunctive Databases. *New Generation Computing* 9(3/4):365–386.

Gelfond, M., and Lifschitz, V. 1998. Action languages. *Electronic Transactions on AI* 3(16):193–210.

Hayes, P. J., and McCarthy, J. 1969. Some Philosophical Problems from the Standpoint of Artificial Intelligence. In Meltzer, B., and Michie, D., eds., *Machine Intelligence 4*. Edinburgh University Press. 463–502.

Marek, V. W., and Truszczynski, M. 1999. *Stable models and an alternative logic programming paradigm*. The Logic Programming Paradigm: a 25-Year Perspective. Springer Verlag, Berlin. 375–398.

Niemelä, I. 1998. Logic Programs with Stable Model Semantics as a Constraint Programming Paradigm. In *Proceedings of the Workshop on Computational Aspects of Nonmonotonic Reasoning*, 72–79.

VanPatten, B. 1993. Grammar teaching for the acquisition-rich classroom. *Foreign Language Annals* 26:435–450.

VanPatten, B. 2002. Processing Instruction: An Update. *Language Learning* 52(4):755–803.

VanPatten, B. 2004. *Input Processing in Second Language Acquisition*. Mahwah, NJ: Lawrence Erlbaum Associates. 5–32.