Generating Pragmatically Appropriate Sentences from Logic: The Case of the Conditional and Biconditional

Renhao Pei and Kees van Deemter

Abstract It is widely assumed that there exist mismatches between the connectives of Propositional Logic and their counterparts in Natural Language. One mismatch that has been extensively discussed is Conditional Perfection, the phenomenon in which a conditional sentence is interpreted as a biconditional under some circumstances. The Pragmatics literature has provided valuable insights into the question of whether Conditional Perfection will happen in a given context. In order to make these insights more explicit and testable, we designed an algorithm to generate pragmatically more appropriate sentences from propositional logical formulas involving material implication and biconditional implication. This algorithm was tested in an evaluation by human participants, in which generated sentences are compared against those generated by a simple baseline algorithm. The evaluation results suggest that the designed algorithm generates better sentences, which capture the semantics of the logical formulas more faithfully.

Keywords Conditional perfection · Propositional logic · Natural language generation

1 Introduction

Mathematical Logic has been extensively used for representing meaning of natural language in Formal Semantics. This is not unproblematic, because a formula in logic is rarely completely equivalent with a sentence of, for example, English. Even in the simplest component of first-order logic (FOL), namely propositional logic, mismatches between the logical connectives and their natural language counterpart are known to exist. These mismatches have long been noticed and studied by logicians

R. Pei (✉) · K. van Deemter
Utrecht University, Utrecht, The Netherlands
e-mail: renhaopei@gmail.com

K. van Deemter
e-mail: c.j.vandeemter@uu.nl

R. Loukanova et al. (eds.), *Logic and Algorithms in Computational Linguistics 2021 (LACompLing2021)*, Studies in Computational Intelligence 1081,
https://doi.org/10.1007/978-3-031-21780-7_8

and linguists, and from their studies, valuable insights about these mismatches have been gained. However, these insights have often lacked the kind of formal detail that makes them testable; consequently, they are difficult to make use of in practical applications.

One relevant area of applications, which we will be looking at in this paper, is Natural Language Generation (NLG) [8, 16]. Of particular relevance is a line of work in which NLG programs take a logical formula as their input, and produce a clear and intelligible English sentence as output. Applications of this kind are of substantial present importance in connection with "explainable Artificial Intelligence", whose aim is to make the workings of Artificial Intelligence programs understandable to human users and other stakeholders (see, e.g., [15]). Mismatches between natural language and logic naturally also play a role in this field.

These two lines of study can complement each other well, as NLG from logic can provide a practical framework for testing theoretical insights while these insights can in turn improve the clarity of natural language sentences generated out of logical formulas.

In the following Sects. 1.1 and 1.2, Condition Perfection and NLG from logic will be explained in more details respectively, which serve as the theoretical background for the rest of the paper.

1.1 Conditional Perfection

The term Conditional Perfection (henceforth referred to as CP) was coined by Geis and Zwicky [9],[1] it refers to the phenomenon that the conditional in the form of *if p, then q* often invites an inference of *if not p, then not q*. Through the mediation of this invited inference, the original conditional will express ('be perfected into') its corresponding biconditional *if and only if p, then q*, as illustrated by the example in [9]:

(1) a. If you mow the lawn, I'll give you five dollars
    b. If you don't mow the lawn, I won't give you five dollars
    c. If and only if you mow the lawn, I'll give you five dollars

Here, the sentence (1a) would also suggest (1b), thus conveying the meaning of (1c). The phenomenon of (1a) inviting the inference of (1b) is called CP.[2]

With propositional logic, Table 1 clearly illustrates how CP is derived through the mediation of an invited inference: Here, the original proposition $p \rightarrow q$ invites an inference of $\neg p \rightarrow \neg q$, and the conjunction of $p \rightarrow q$ and $\neg p \rightarrow \neg q$ will have the

[1] Geis and Zwicky [9] coined the term CP, but they were not the first to address this phenomenon in linguistics. See [2] for a historic overview.

[2] There have been two different usages of the term CP, the first one is that the invited inference of (1b) constitutes CP, the other one is that the biconditional in (1c) constitutes CP. In this paper, the term CP is used to refer to the invited inference itself.

Table 1 Truth table illustrating how CP is derived from an invited inference

p	q	p → q	¬p → ¬q	(p → q) ∧ (¬p → ¬q)	p ↔ q
T	T	T	T	T	T
T	F	F	T	F	F
F	T	T	F	F	F
F	F	T	T	T	T

same truth table as $p \leftrightarrow q$. As a result, the original material implication $p \rightarrow q$ is perfected into a biconditional implication $p \leftrightarrow q$.

Various explanations have been proposed concerning the nature of CP. Van der Auwera [1] regarded CP as a scalar implicature. In that approach, CP invokes the scale ⟨(*if p, q* and *if r, q* and *if s, q*), (*if p, q* and *if r, q*), (*if p, q*)⟩. The assertion of *if p, q* then implicates there is no other antecedent (*r*, *s*, etc.) with *q* as a consequent, thus expressing *if and only p, q*.

A second approach proposed by Horn [11] considers CP as an implicature motivated by the R-based pragmatic strengthening. In this approach, CP is the result of the second Maxim of Quantity (*do not make your contribution more informative than is required*) and the Maxim of Relation (*be relevant*). As the stipulated condition *p* in *if p, q* must be taken to be informative and relevant, *p* would be interpreted as necessary, hence the original sufficient condition is strengthened into a necessary and sufficient condition.

Thirdly, Herburger [10] proposed the 'whole truth theory' to account for CP. In this proposal, a sentence *S* can be silently conjoined with *only S*, resulting in the conjunction *S and only S*, which is then taken to express 'the truth and the whole truth'. Therefore, a conditional sentence with *if* will be taken to express *if and only if.*

In the present study, no position regarding the nature of CP is taken. Whatever the explanation of CP might be, for our purpose of building an NLG system, what matters most is the distribution of CP, i.e., when CP is expected to occur, and when CP is expected to not occur.

While the above-mentioned studies sought to propose a theory about the nature of CP, Van Canegem-Ardijns and Van Belle [18] sought to provide a descriptive account for CP. Van Canegem-Ardijns and Van Belle [18] identified three sub-types of CP: two specific ones (*only if p, q and only if not p, not q*), and a more general one (*if not p, then not q*), and showed the correlation between each of the three sub-types of CP with various speech act or utterance types, as well as some utterance types in which CP will not occur. Among the many utterance types that [18] discussed, three of them are particularly relevant for the purpose of the present study, namely: conditional promise, conditional threat and conditional warning.

According to [18], in a conditional promise, the speaker attempts to get the hearer to do something by offering a potential reward. It involves examples as 'If you get me some coffee, I'll give you a cookie.' Conditional promises have the following semantic characteristics: [*A is desirable for S; H has control over A; S wants H to do A; S considers C desirable for H because S assumes H wants C; S has control over C*] (Here the abbreviations follow the usage in [18]: A = action/event described in the antecedent, C = action/event described in the consequent, S = speaker, H = hearer).

In a conditional threat, the speaker attempts to refrain the hearer from doing something by threatening with a potential consequent, such as 'If you lie to me, I'll kill you.' Conditional threats have the following semantic characteristics: [*A is undesirable for S; H has control over A; S wants H to do not-A; S considers C undesirable for H because S assumes H does not want C; S has control over C*]. It is easy to see that the conditional threat and the conditional promise are parallel to each other, the only difference is that the antecedent and the consequent in a conditional promise are both desirable while they are both undesirable in a conditional threat.

The conditional warning resembles the conditional threat in many ways, as they both share the semantic characteristics of [*H has control over A; S wants H to do not A; S considers C undesirable for H because S assumes H does not want C*]. One crucial difference that sets them apart is that the conditional warning lacks the semantic characteristic of [*S has control over C*]. For instance, in the conditional warning of 'If you park your car on private land, you will get your car wheel clamped,' the speaker does not have actual control over whether the car wheel will be clamped or not.

According to [18], conditional promises readily invite an inference of *only if p, q* (*q* in no event other than that *p*), which entails *if not p, then not q*. Conditional threats invite an inference of *only if not p, not q*, which also entails *if not p, then not q*. On the other hand, conditional warning does not invite the inference of *only if not p, not q*, since the speaker does not have control over the consequent in a conditional warning, she or he cannot justifiably guarantee whether the consequent will or will not take place. Van Canegem-Ardijns and Van Belle [18] did not explicitly mention whether conditional warning invites the more general inference of *if not p, then not q*, while several experimental studies involving human judgements [5, 13, 14] have shown that, conditional warning is indeed less likely to invite the inference of *if not p, then not q*, compared to conditional threats and conditional promises.

In these experimental studies [5, 13, 14], apart from the already mentioned conditional promise, conditional threat and conditional warning, a fourth type of conditionals (or utterance type as in Van Canegem-Ardijns and Van Belle[18]'s terminology) is involved, which is conditional tip. An example of conditional tip could be 'If you show up early for work, you will impress your boss,' in which the consequent is desirable for the hearer, while the speaker does not have control over the consequent. The conditional tip resembles conditional promise, in the same way that conditional warning resembles conditional threat, thus forming a complete tetrachoric table as in Table 2.

Conditional promise and conditional threat can be grouped together under the umbrella term *inducement*, as their utterances offer inducements to alter people's behavior and only differ in that conditional promise encourages actions by potential

Table 2 Classification of conditionals into four sub-types using two features

	Speaker having control over the consequent	Speaker **not** having control over the consequent
The consequent being desirable for the hearer	Conditional promise	Conditional tip
The consequent being **un**desirable for the hearer	Conditional threat	Conditional warning

desirable reward while the conditional threat deters actions by prospective undesirable punishment. On the other hand, conditional tip and conditional warning can be grouped as *advice*, as their utterances offer advice and again only differ with regard to whether the outcome is desirable or undesirable.

Note that in Table 2, both features used for the classification are about consequent, while for the antecedent, whether the antecedent is desirable and whether the hearer has control over the antecedent are not mentioned. Similarly, in [18] the desirability of the antecedent is always in accordance with the desirability of the consequent and the hearer always has control over the antecedent. In the present study, we will follow Table 2 and omit the information whether the hearer has control over the antecedent and whether the antecedent is desirable, since this kind of information can be easily derived and is therefore redundant. The fact that only the consequent plays a role in classification of the conditionals will be reflected in the design of the algorithm discussed in Sect. 2.

Despite different experimental settings, all of the three studies [5, 13, 14] have shown that inducements tend to invite the inference of *if not p, then not q*, while advice is significantly less likely to invite such an inference. In other words, CP is more likely to occur in inducements and less likely to occur in the advice.

1.2 Generating Natural Language from Logical Formulas

Natural Language Generation (NLG) refers to the domain of generating natural language text from some non-linguistic representations of information. For example, a weather forecast NLG system can generate weather forecast in text from raw meteorological data. The present study, however, utilizes a different source of information, namely the logical formula, as the input for an NLG system.

Logic is a useful tool in many different domains. It has been extensively used for representing meaning in the Formal Semantics, as well as for representing argument in formal argumentation. While logical formula has the advantage of being formal and clearly defined, it is certainly not the most understandable format for representing information, since not everyone knows logic.

For example, as has been put forward in [15], when an automatic system is asked to give explanations to a human operator for why it takes a certain action or why it provides a certain plan for the user, then the best that many systems can do is to respond in the form of logical formula, which is however opaque for most non-technical human users. Mayn and Van Deemter [12] have also remarked that in Explainable AI, there exists a focus on making algorithms in Artificial Intelligence transparent. For that purpose, tools that can produce automatic English 'translations' from logical formulas can offer a good way to convey information to users, especially if these users are not familiar with Mathematical Logic. In this case, an NLG system that can translate logical formula into natural language is needed to make the autonomous system scrutable to non-technical users.

The task of NLG from logical forms dates from 1980s [20] and various kinds of logical forms and different approaches have been used, targeting at different practical applications.

Among them, one that is particularly relevant to the present study is the NLG system of [7], which provides feedback to students of logic. The logic form used as input in [7] is the first-order logic (FOL), which is one of the most basic forms of logic and is widely used in domains such as Artificial Intelligence, Linguistics, and Philosophy. FOL is also the logic most often taught [4, 6]. When students learn the FOL, one crucial exercise is to translate between natural language sentences and their corresponding FOL representations. For this kind of translation exercises, an automated NLG system could greatly facilitate a student's learning process by generating natural language sentences from the student's incorrect solution in FOL. This generated natural language sentence could show the student what his or her proposed solution actually says, as a way to prompt repair.

However, to build such an NLG system is not a trivial task. One important obstacle for translating between natural language and FOL faithfully is that there are many mismatches between the two. Even in the simplest component of FOL, namely the propositional logic, mismatches between the logical connectives and their natural language counterpart exist, such as the phenomena *Conditional Perfection* and *exclusive/inclusive disjunction*. These issues have long been noticed and studied by logicians and linguists, and from their studies, valuable insights have been gained, which could then be used to help building a successful NLG system.

The present study focuses on the Conditional Perfection. As explained in Sect. 1.1, the ideas drawn from theoretical and experimental studies about Conditional Perfection will then be used to improve an NLG system that deals with the two logical connectives $\rightarrow$ and $\leftrightarrow$.

1.3 Focus of the Paper

Under the general topic of testing the insights gained from Pragmatics literature in the framework of NLG from logic, the present paper will revolve around how to generate more appropriate sentences to deal with CP.

For the input of propositional logical formulas in the present study, only two logical connectives are included, namely $\rightarrow$ for material implication and $\leftrightarrow$ for biconditional implication, as CP is only relevant to these two connectives. Notably, [3] reported that students of logic were found to have difficulty with distinguishing these two connectives, and the errors caused by confusing them account for 29.77% of total error in the corpus of student translations of natural language into logic. Since this type of error is so common, it would be very helpful to build an automatic program that could show the students what their errored logical formulas containing $\rightarrow$ and $\leftrightarrow$ really say in natural language. Therefore, a simple NLG system that only focuses on these two connectives would already have great merits for practical purpose.

What's more, the present study only uses binary propositions, i.e., there is only one connective for each propositional formula, and the formula would be in the form of [literal[3] + connective + literal]. This is only for convenience's purpose and to avoid potential interactions caused by multiple connectives within one proposition. Among various kinds of conditionals, the present study only focuses on 4 types of conditionals namely promise, threat, warning, and tip, as described in Sect. 1.1. And lastly, the natural language used in this study is restricted to English.

2 A Pragmatic Algorithm for Expressing Propositional Logic Formulas in English

2.1 Key Ideas Underlying the Algorithm

As explained in Sect. 1.1, the 4 types of conditionals under discussion (promise, threat, warning, and tip) are distinguished by two features: 1. Whether the speaker has control over the consequent, and 2. Whether the consequent is desirable or undesirable for the hearer. These two features, each having two values (positive or negative), can be abbreviated as [±control] and [±desirability].

At the first stage of the algorithm, a set of binary logical propositions is randomly generated by firstly selecting one connective from either $\rightarrow$ or $\leftrightarrow$, and then selecting one antecedent and one consequent from two pre-defined atomic proposition banks (an antecedent bank and a consequent bank) respectively. Each atomic proposition in the consequent bank is associated with its features of [±control] and [±desirability], while the atomic propositions in the antecedent bank are not associated with these two features. As explained in Sect. 1.1, this is because the antecedents do not play any role in distinguishing the 4 types of the conditionals. In all the 4 types of conditionals under discussion, the hearer would always have control over the antecedent, i.e., the antecedent would always be [+control], thus making this information redundant. In terms of desirability, the atomic propositions for antecedents are intentionally constructed to be inherently neutral, so that within a binary proposition, the inher-

[3] A literal is an atomic formula p or its negation $\neg p$.

Table 3 Truth table showing the effect of cancelling CP

p	q	p → q	If p, q (inducement)	If p, q, but if not p, might still q
T	T	T	T	T
T	F	F	F	F
F	T	T	F	T
F	F	T	T	T

ently neutral antecedent could be interpreted to be either desirable or undesirable in accordance with the consequent.

By utilizing the information from features of the consequent, each proposition can then be classified as either a promise, a threat, a warning, or a tip. CP is expected to occur in inducements (promises and threats) and not occur in the advice (warnings and tips). After having classified the conditionals into the 4 types, we can have a prediction of whether CP will occur or not.

For the input binary proposition of material implication p → q, the most straightforward way is to translate the connective as 'if p, then q.' But if the sentence is an inducement, then CP will occur, which is not preferable since it would be a distortion of the original meaning of the logical formula, and we want the NLG system to faithfully express the logical meaning in natural language. Fortunately, when CP is present, it can still be cancelled by adding an additional condition as noted by Herburger [10]. As in the following example:

(2) a. If you buy him a drink, I will help you
    b. If you buy him a drink, I will help you, and if you don't, I might still do

Here the sentence (2a) is a conditional promise and would invite an inference of *if you don't buy him a drink, I won't help you*, thus expressing *if and only if you buy him a drink, I will help you*. However, this inference can be cancelled by adding and if you don't, I might still do as in (2b). The meaning of p → q (with p being 'you buy him a drink' and q being 'I will help you') is faithfully preserved in (2b), but not in (2a), in which CP occurs. As shown in the Table 3.

Therefore, for the binary proposition in the form of p → q, if the proposition is classified as an inducement, then CP is expected to occur and would be translated as 'if p, q, but if not p, might still q.' On the other hand, if p → q is advice, then CP is not expected to occur and there is no need to add 'but if not p, might still q.'

As for the biconditional implication p ↔ q, the most straightforward way is to translate it as 'if and only if p, q.' But if the proposition is classified as an inducement, then CP is expected to happen, which means 'if p, q' would express the same meaning as 'if and only if p, q.' In this case, we can simply use 'if p, then q' for expressing p ↔ q, which would be a better choice of words, since 'if and only if' is not a very natural expression and people typically do not use this phrase outside the domain of mathematics or logic. On the other hand, if p ↔ q is advice, then CP will not happen and we have no other choice but to use ‘if and only if,’ if we want to faithfully preserve the meaning of the original logical formula.

2.2 The Pragmatic Algorithm and the Baseline

In this section, to demonstrate what the designed algorithm really does, a comparison will be made between two algorithms: one so called ‘pragmatic algorithm’ that tries to generate pragmatically more appropriate sentences following the ideas explained in Sect. 2.1, and another baseline algorithm that does nothing pragmatic and only do the most straightforward realization.

Below are some pseudo-codes illustrating the baseline algorithm, in which l is a logical formula that is in the form of either $p \rightarrow q$ or $p \leftrightarrow q$, and r is the realized sentence for the logical formula.

Algorithm 1 The baseline algorithm

```
if $l$ is in the form of $p \rightarrow q$ then
    $r \leftarrow$ ‘If p, q.’
else if $l$ is in the form of $p \leftrightarrow q$ then
    $r \leftarrow$ ‘If and only if p, q.’
end if
```

The input for the baseline algorithm is a list of randomly generated binary logical formulas that are in the form of either p $\rightarrow$ q or p $\leftrightarrow$ q. If the logical formula l is a material implication (i.e., in the form of p $\rightarrow$ q), it will be realized into ‘If p, q’. If the logical formula l is a biconditional implication (in the form of p$\leftrightarrow$ q), it will be realized into ‘If and only if p, q’.

What the baseline algorithm essentially does here, is simply translate the logical connectives $\rightarrow$ and $\leftrightarrow$ into their commonly used natural language counterparts ‘if…then’ and ‘if and only if…then’, while the pragmatic algorithm does a more sophisticated realization.

The pragmatic algorithm also takes the list of logical formulas as an input. Besides, it also takes in a label dictionary, which contains the information about the corresponding label for each logical formula. These labels include *promise*, *threat*, *tip*, and *warning*, and furthermore, *promise* and *threat* are under the umbrella label *inducement*, while *tip* and *warning* are under the label *advice*. These labels are automatically obtained by a label detector that can label a binary proposition according to the predefined features ([±control] and [±desirability]) of the consequent, with the rationale explained in Sect. 2.1 and Table 2.

Below are some pseudo-codes illustrating the pragmatic algorithm, with a basic setting similar to Algorithm 1.

The pragmatic algorithm realizes the logical formulas differently depending on their labels. Compared to the baseline algorithm, the improvement lies in the situation

Algorithm 2 The pragmatic algorithm

```
if $l$ is in the form of $p \rightarrow q$ then
    if $l$ has the label ‘inducement’ then
        $r \leftarrow$ ‘If p, q, but if not p, might still q.’
    else if $l$ has the label ‘advice’ then
        $r \leftarrow$ ‘If p, q.’
    end if
else if $l$ is in the form of $p \leftrightarrow q$ then
    if $l$ has the label ‘inducement’ then
        $r \leftarrow$ ‘If p, q.’
    else if $l$ has the label ‘advice’ then
        $r \leftarrow$ ‘If and only if p, q.’
    end if
end if
```

where the logical formula is labeled as ‘inducement’. If a logical formula of material implication p $\rightarrow$ q is labeled as ‘inducement’, some extra words ‘but if not p, might still q’ will be appended after the basic sentence ‘If p, q’, in order to cancel the expected CP. If a logical formula of biconditional implication p $\leftrightarrow$ q is labeled as ‘inducement’, the connective will be translated as ‘If…then’ rather than ‘If and only if…then’, in order to make the realized sentence shorter and more natural. On the other hand, when the logical formula is labeled as ‘advice’, the pragmatic algorithm will do exactly the same realization as the baseline algorithm does.

2.3 Example of Input and Output

Before showing the input and output example, it is important to note that although the algorithm can inherently work for any proposition of promises, threats, tips, and warnings, it also has some other general requirements in practice. Since the binary logical propositions are all generated by randomly selecting one antecedent and one consequent, it is important to ensure that all the random combinations of an antecedent and a consequent make sense. Not any two atomic propositions can form a good proposition in which the connection between the antecedent and the consequent can be easily inferred through common sense. A bad example (in natural language) could be ‘If you drink this bottle of milk, I will book a ticket to Iceland.’ The connection between drinking milk and booking a ticket is hard to establish, thus making this sentence sound weird. To ensure that all the generated binary propositions make sense, a good practice is to limit the atomic propositions within some specific domain.

This domain of propositions in our example, is the communication between players in a strategic video game, in which promises, threats, tips, and warnings could often be made. Suppose we have three atomic propositions: 1. *You destroy the bridge*, 2. *I will attack you* and 3. *Player C will attack you*, in which the first one is antecedent and the latter two are consequents. Here, the first proposition is inherently neutral regarding desirability. The second and the third propositions are both undesirable for the hearer, and they differ in that the speaker has control in *I will attack you*, but lacks control in *Player C will attack you*. Together with two connectives $\rightarrow$ and $\leftrightarrow$, four binary logical formulas can be generated:

(3) a. You destroy the bridge $\rightarrow$ I will attack you
 b. You destroy the bridge $\rightarrow$ Player C will attack you
 c. You destroy the bridge $\leftrightarrow$ I will attack you
 d. You destroy the bridge $\leftrightarrow$ Player C will attack you

Based on the information of control and desirability of the consequent, they can be easily labeled as:

(4) a. You destroy the bridge $\rightarrow$ I will attack you:'inducement(threat)'
 b. You destroy the bridge $\rightarrow$ Player C will attack you:'advice(warning)'
 c. You destroy the bridge $\leftrightarrow$ I will attack you:'inducement(threat)'
 d. You destroy the bridge $\leftrightarrow$ Player C will attack you:'advice(warning)'

Taking a list containing the four formulas in (3) as the input, the baseline algorithm will yield:

(5) a. If you destroy the bridge, I will attack you
 b. If you destroy the bridge, player C will attack you
 c. If and only if you destroy the bridge, I will attack you
 d. If and only if you destroy the bridge, player C will attack you

With a list containing the four formulas in (4), alongside with the information about the corresponding labels as input, the pragmatic algorithm will yield output as:

(6) a. If you destroy the bridge, I will attack you, but if you don't, I might still do
 b. If you destroy the bridge, player C will attack you
 c. If you destroy the bridge, I will attack you
 d. If and only if you destroy the bridge, player C will attack you

Comparing (6) with (5), the advantage of the pragmatic algorithm can be clearly seen. While both algorithms take the list of logical propositions as input, the pragmatic algorithm yield more appropriate natural language sentences as output by also taking the extra information from the labels. (6a) would express the original semantics of the logical formula more faithfully than (5a), and (6c) would sound more natural compared to (5c), while (6c) and (5c) convey the same meaning.

3 Evaluating the Algorithm

3.1 Design and Materials

As shown in Sects. 2.2 and 2.3, the pragmatic algorithm is designed to be a better algorithm than the baseline algorithm. For material implication, the pragmatic algorithm should express the logical meaning more faithfully by adding some extra phrases to cancel CP. For biconditional implication, the pragmatic algorithm should generate more natural sentences by using 'if…then' instead of 'if and only if…then', while also preserving the logical meaning faithfully. On the other hand, we also hypothesis that the extra phrases added for of material implication would make the sentences longer and therefore slightly more unnatural, as the cost of making the sentences more faithful.

But so far, the merits of the pragmatic algorithm have been purely theoretical. For validation, an evaluation is made by asking human participants to evaluate the natural language output from both algorithms. The evaluation has two metrics: naturalness and faithfulness.

To evaluate the naturalness of the output, the participants are asked to mark a number on a linear scale from 1 to 5 (from 'very unnatural' to 'very natural') for how natural they think the generated natural language sentence sounds.

To evaluate whether generated natural language sentence faithfully conveys the original semantic meaning of the logical formula, an adapted version of *truth table task* is used. As in [17], the *truth table task* refers to the task in which 'participants are given a *rule* and asked to indicate which cases are consistent with that *rule*.' In our adapted version, this task is placed within the setting of the communication between players in a hypothetic strategy game,[4] as described in Sect. 2.3. The *rule* comes in the form of a message sent by one player to another player, and the participant are asked to select all the cases where both players' actions are true and consistent with that message.

To prepare the cases in the checkbox for each logical formula, the antecedent and the consequent in the original formula can be either true (unchanged) or false (negated), and then they are combined through conjunction. Thus, 4 cases (TT, TF, FT, FF) will be automatically created for each logical formula. For example, from a logical formula like *You destroy the bridge* $\rightarrow$ *I will attack you*, the four cases are: *You destroy the bridge* $\wedge$ *I will attack you* (TT), *You destroy the bridge* $\wedge$ $\neg$*I will attack you* (TF), $\neg$*You destroy the bridge* $\wedge$ *I will attack you* (FT), $\neg$ *You destroy the bridge* $\wedge$ $\neg$*I will attack you* (FF). These four cases will be realized into natural language sentences, and then be presented in a random order within each question.

Below is an example question demonstrating how the truth table task is used in the evaluation:

[4] The settings of this strategy game are very intuitive and should be easily understood through common sense (See Appendix for the game settings). The settings are presented to the participants before the evaluation begins, to ensure that all the participants will have a unified world knowledge about how the game works.

In a game, one player Sophie sent a message to another player Hans: 'If you destroy the bridge, I will attack you, and if you don't, I might still do.'
Having received the message,
(Tick all that apply)

☐ *Hans didn't destroy the bridge, and Sophie attacked him*
☐ *Hans destroyed the bridge, and Sophie attacked him*
☐ *Hans destroyed the bridge, and Sophie didn't attack him*
☐ *Hans didn't destroy the bridge, and Sophie didn't attack him*

If the participant ticks the checkbox of a particular case, it means that the participant judges this case to be true for the message, and if the participant leaves the box unchecked, it means that the participant judges this case to be false for the message. Similar to the naturalness score, a faithfulness score can be derived, through comparing the participant's answers with the truth table value of the original logical formula.

For instance, according to the truth table value of the material implication *You destroy the bridge* $\rightarrow$ *I will attack you*, if the participant's understanding of the generated natural language message 'If you destroy the bridge, I will attack you, and if you don't, I might still do.' is the same as the meaning of the logical formula, the checkbox would be ticked as:

☑ *Hans didn't destroy the bridge, and Sophie attacked him*
☐ *Hans destroyed the bridge, and Sophie attacked him*
☑ *Hans destroyed the bridge, and Sophie didn't attack him*
☑ *Hans didn't destroy the bridge, and Sophie didn't attack him*

If the participant leaves any checkbox unticked where it should be ticked, or ticks any checkbox which should not be ticked, it would mean that the participant has a different understanding of the generated natural language sentence, as compared to the semantics of the logical formula, thus making the natural language generation unfaithful. If the judgment of the participant is completely in accordance with the truth table value of the original logical formula, then the participant can receive 4 points. For each checkbox containing a different value than the truth table of the original logical formula, 1 point would be deducted, thus making the faithfulness score in a range of 0–4. The greater the difference, the lower the faithfulness score will be.

Since the pragmatic algorithm and the baseline algorithm yield the same output for advice (tips and warnings), these identical sentences are naturally not included in the evaluation, as the purpose of the evaluation is to see whether the outputs of these two algorithms are evaluated differently in terms of naturalness and faithfulness. For each algorithm, the evaluation material includes both promise and threat, and for each type, one conditional and one biconditional are included. Therefore, the evaluation form has 2 (baseline and pragmatic) $\times$ 2 (promise and threat) $\times$ 2 (conditional and biconditional) = 8 target sentences, plus 8 filler sentences, hence 16 sentences in total. The filler sentences are some conditionals containing disjunction and conjunction, and a concessive conditional with 'Even if', which are not the topic of this present

Table 4 Mean naturalness score comparison between baseline algorithm and pragmatic algorithm

	Baseline algorithm	Pragmatic algorithm
Material implication	4.54	3.35
Biconditional implication	3.95	<u>4.6</u>

study. Each sentence in the evaluation form has 2 evaluation tasks: (a) a truth table task for evaluating faithfulness and (b) a linear scale for evaluating naturalness.

3.2 Participants and Procedure

40 participants[5] took part in the evaluation. Unlike grammatical judgment, this evaluation relies more on conscious thinking and less on linguistic intuition, therefore being a native speaker of English was not required. Nonetheless, all participants were proficient in English (having degree courses taught in English and/or being native English speaker).

The evaluation was done online by asking participants to firstly fill in their personal information of gender, age, and English proficiency, and to read the instructions, and then to fill in the actual evaluation form. No time limit was posed for the evaluation.

3.3 Results and Discussion

The **naturalness score** results confirmed the theoretically motived hypothesis stated in Sect. 3.1, as shown in Table 4: For biconditional implication, the pragmatic algorithm achieved a higher mean naturalness score 4.6 compared to the 3.95 of the baseline algorithm. To test the statistical significance of this difference, a paired t-test is performed, showing a statistical significance between the naturalness scores for biconditional implication of baseline ($M = 3.95$, $SD = 1.03$) and that of the pragmatic algorithm ($M = 4.6$, $SD = 0.61$), $t(39) = 6.00$, $p < 0.001$. On the other hand, for the material implication, the mean naturalness score of the pragmatic algorithm is only 3.35, less than the 4.54 of the baseline algorithm. This difference is also tested using a paired t-test, also showing a statistical significance between the naturalness score for material implication of baseline ($M = 4.54$, $SD = 0.65$) and that of the pragmatic algorithm ($M = 3.35$, $SD = 1.13$), $t(39) = 8.76$, $p < 0.001$. Therefore, the evaluation results show that the pragmatic algorithm generates more natural sentences for biconditional implication. It generates less natural sentences

[5] Of the 40 participants, 20 were male, 18 were female, and 2 indicated as ‘other’ in the form. The average age of the participants was 37.6 years.

Table 5 Mean faithfulness score comparison between baseline algorithm and pragmatic algorithm

	Baseline algorithm	Pragmatic algorithm
Material implication	2.9	<u>3.44</u>
Biconditional implication	3.56	<u>3.78</u>

Table 6 Truth table of material implication and biconditional implication

p	q	$p \rightarrow q$	$p \leftrightarrow q$
T	T	T	T
T	F	F	F
F	T	T	F
F	F	T	T

for material implication, but it can be argued that their lack of naturalness is offset by their substantially enhanced faithfulness, as we shall see.

For the **faithfulness score**, the means are shown in Table 5. The results show that the pragmatic algorithm has achieved a higher mean faithfulness score for both material implication and biconditional implication. For both material implication and biconditional implication, the paired t-test is performed to test whether the difference has statistical significance. The result indicates that there is a statistical significance between the faithful scores for material implication of baseline ($M = 2.9$, $SD = 0.44$) and that of the pragmatic algorithm ($M = 3.44$, $SD = 0.74$), $t(39) = 6.60$, $p < 0.001$. And there is also a statistical significance between the faithful scores for biconditional implication of baseline ($M = 3.56$, $SD = 0.74$) and that of the pragmatic algorithm ($M = 3.78$, $SD = 0.48$), $t(39) = 2.76$, $p = 0.007$.

These results show that the pragmatic algorithm can express the logical meaning more faithfully for both material implication and biconditional implication. This confirmed our expectation about material implication, while exceeding our expectation about biconditional implication. As prior to the evaluation, the pragmatic algorithm is designed to only improve the naturalness for biconditional, rather than being designed to improve the faithfulness. But the evaluation results suggest that 'if…then' is not only more natural, but also expresses the biconditional connective more faithfully compared to 'if and only if…then'.

Apart from the faithfulness scores and naturalness scores calculated for each question, we can take a closer look at the results of the all the four cases (TT, FF, FT, FF) in the truth table task separately. For each of the four rows in the truth table, if the participant's judgement is the same as in the truth table of the logical connectives, it will be interpreted as the semantic meaning is faithfully convey to the participant, otherwise, it means the meaning is conveyed unfaithfully (Table 6).

Table 7 show the results of material implication for both algorithms. The numbers in the columns of TT, TF, FT, FF stand for the accuracy score for each case

Table 7 Results of accuracies of TT, TF, FT, and FF in truth table task for material implication

	p → q				
	TT	TF	FT	FF	Average accuracy
Baseline algorithm	0.99	0.95	<u>0.08</u>	0.86	0.72
Pragmatic algorithm	0.95	0.83	<u>0.88</u>	0.79	0.86

Table 8 Results of accuracies of TT, TF, FT, and FF in truth table task for biconditional implication

	p ↔ q				
	TT	TF	FT	FF	average accuracy
Baseline algorithm	0.94	0.89	<u>0.9</u>	0.84	0.89
Pragmatic algorithm	0.99	0.94	<u>0.94</u>	0.91	0.94

separately, and the *average accuracy* is the average score of all four cases. All the numbers presented are averaged across all participants.

The results show that the participant generally had high accuracies for both algorithms in the cases of TT, TF and FF, showing a good understanding of the semantics of $p \to q$ in these three cases. However, the sentences generated by the baseline algorithm seems to pose a great difficulty for participant in the case of FT, scoring an accuracy of only 0.08, which means for 92% of the time the participants believe $p \to q$ should be false in the case of FT. This is exactly what we predict, because of the effect of CP.

On the hand, the pragmatic algorithm can greatly improve the accuracy from 0.08 to 0.88, result in a higher average accuracy of 0.86, compared to the 0.72 of the baseline algorithm. This shows that the pragmatic algorithm has a great advantage for faithfully expressing the semantics of $p \to q$, especially for the FT case.

Similar to the Tables 7, 8 shows the results for biconditional implication, that both algorithms have achieved high accuracies for biconditional implication across the four cases of TT, TF, FT, and FF. The accuracies of the pragmatic algorithm are slightly higher, which is in line with the results of faithfulness scores.

To summarize, the evaluation results have shown that for material implication, the pragmatic algorithm can indeed generate sentences that are more faithful at the cost of being less natural. And for biconditional implication, the sentences generated by the pragmatic algorithm are both more faithful and more natural compared to the baseline algorithm. All these differences between the pragmatic and baseline algorithm are statistically significant.

4 Conclusion and Future Work

In this study of Conditional Perfection, we started from insights drawn from the pragmatic literature [5, 13, 14, 18]. Although these insights are interesting and plausible, they are also less than fully explicit, and have never been formally evaluated or implemented in an algorithm that expresses logical information in natural language. In the present work, we have provided an algorithmic formalisation of some pragmatic insights and subjected these to experimental testing with human subjects.

Inspired by the insight that Conditional Perfection (CP) is more likely to happen in *inducements* and less likely to happen in *advice*, an algorithm was designed to generate pragmatically more appropriate sentences from propositional logical formulas involving the two connectives $\rightarrow$ and $\leftrightarrow$. The merits of our algorithm have been tested in an evaluation by human participants, in which the sentences generated by the pragmatic algorithm are compared with the sentences generated by a simple baseline algorithm.

The theoretically motivated expectations behind the pragmatic algorithm have been confirmed in the evaluation. The results suggested that, by adding as 'if p, q, but if not p, might still q,' the effect of CP can be canceled, resulting in a great improvement for the FT case in the truth table (the row where the antecedent is false and consequent is true), thus improving the faithfulness of the sentence. The evaluation results also confirmed that by using 'If…then' instead of 'If and only if…then', the pragmatic algorithm improves the naturalness of the generated sentences.

Furthermore, the evaluation results indicated that the pragmatic algorithm using 'If…then' instead of 'If and only if…then' is not only more natural, but also more faithful. This suggests that, although 'If and only if…then' is conventionally used as the natural language counterpart for $\leftrightarrow$, it actually leads to more misunderstanding compared to simply using 'If…then' for $\leftrightarrow$. A possible explanation is the fact that 'if and only if' is not really a common expression outside logic and mathematics, and it may cause confusion to people who are not very familiar with logic or mathematics.

As expected, the pragmatic algorithm also has a disadvantage when it comes to the naturalness of the sentences involving material implication. Saying 'if p, q, but if not p, might still q' boosts the faithfulness, but it also makes the sentence longer and a bit stilted.

As summarized in Table 9 (the symbol '$>$' represents 'surpass', 'being better than'), we can see that the pragmatic algorithm is superior to the baseline algorithm in three ways and is inferior in only one way. In practical NLG applications such as providing automatic feedback to the translation between FOL and natural language (see Sect. 1.2), faithfulness tends to be more important than naturalness. Therefore, it seems fair to conclude that our pragmatic NLG algorithm would be a worthwhile alternative for use in practical NLG applications, for example when NLG is used to clarify logical expressions in Artificial Intelligence applications [15].

Our study has addressed two problems: (1) under what precise circumstances does Conditional Perfection arise, and (2) how an NLG algorithm might optimize the wording of the natural language expression to take Conditional Perfection into

Table 9 Summarized comparison between pragmatic and baseline algorithm

	Biconditional implication	Material implication
Faithfulness	pragmatic > baseline	Pragmatic > baseline
Naturalness	pragmatic > baseline	Baseline > pragmatic

account. We believe that this work paves the way for a number of further investigations. First, the present study has only considered four types of conditionals namely promises, threats, tips, and warnings, as the relations between them are systematic and they can form a complete tetrachoric table through two features. In the future, the question of how to generate pragmatically appropriate sentences for other types of conditionals such as background conditionals and concessive conditional (which have also been discussed in [18]) should be further investigated to see whether they can be handled along the lines of the present paper.

Second, the present study has only dealt with the two logic connectives $\rightarrow$ and $\leftrightarrow$. Future work could be done to investigate other connectives such as $\vee$ and $\wedge$, thus covering all the connectives used in propositional logic. An important question for further research is to what extent our findings are sensitive to the choice of connectives. For example, $\neg p \rightarrow q$ and $p \vee q$ are logically equivalent, but these two logically equivalent formulations may not be pragmatically equivalent. Other constraints governing the natural language form will need to be taken into consideration. For example, a conditional of the form $\neg p \rightarrow q$ can make both promises and threats, whereas the logically equivalent disjunction $p \vee q$ can only make threats, but not promises [19]. If more connectives are to be added into the NLG system, then these issues should be taken into account.

Thirdly, the present study has only considered the binary proposition in which two atomic propositions are connected through a single connective. In future work, more complex propositions involving multiple connectives can be tested, to see if the conclusion based on binary propositions still hold when the proposition becomes more complex. This direction could also lead to further investigating of the interaction between different kinds of connectives when they are present within the same complex proposition.

Finally, it would be worth investigating how our work may be extended to enhance existing Natural Language Processing work in paraphrasing and text simplification [21]. The idea would be to take an input text T that contains conditional statements, and to convert it into a clearer and more explicit output text T' in which these conditionals are rendered in a more explicit way, for instance by replacing conditionals by biconditionals if and when this is appropriate, analogous to our pragmatic NLG algorithm. To make this work, an NLP algorithm would first have to detect whether the relevant texts in T express promises, threats, and so on because, as we have seen, these pragmatic factors affect the way in which conditionals are interpreted.

Appendix: Instructions for Participants in our Experiment

This study is about testing people's understanding about some sentences. The target sentences will be embedded in the setting of the communication between players in a strategy game.

In case you haven't played any strategy game, this is a multiplayer game, in which buildings like castles and bridges can be built to gain advantage. Players can attack other players and destroy other players' buildings to beat them in the game. Players can also form alliance with each other against a common enemy. Resources (including food, wood, and stone) and gold coins are valuable in the game and can be exchanged among players.

In one game, one player Sophie has sent some messages to another player Hans. In the following sections, each message will be accompanied with 4 scenarios.

After you have read the sentence, please identify all the possible scenarios in which the actions of both Sophie and Hans are consistent to the message. In other words, if you think the scenario follows the message, tick the box; if you think the scenario is impossible given the message of Sophie, don't tick the box.

Afterwards, please rate the wording of the message in terms of how natural the message sounds to you.

References

1. Van der Auwera, J.: Conditional perfection. Amst. Stud. Theory Hist. Linguist. Sci. Ser. **4**, 169–190 (1997)
2. Van der Auwera, J.: Pragmatics in the last quarter century: the case of conditional perfection. J. Pragmat. **27**(3), 261–274 (1997)
3. Barker-Plummer, D., Cox, R., Dale, R., Etchemendy, J.: An empirical study of errors in translating natural language into logic. In: Proceedings of the Annual Meeting of the Cognitive Science Society, vol. 30 (2008)
4. Enderton, H.B.: A Mathematical Introduction to Logic. Elsevier (2001)
5. Evans, J.S.B., Twyman-Musgrove, J.: Conditional reasoning with inducements and advice. Cognition **69**(1), B11–B16 (1998)
6. Fitting, M.: First-Order Logic and Automated Theorem Proving. Springer Science & Business Media (2012)
7. Flickinger, D.: Generating English paraphrases from logic. In: From Semantics to Dialectometry (2016)
8. Gatt, A., Krahmer, E.: Survey of the state of the art in natural language generation: core tasks, applications and evaluation. J. Artif. Intell. Res. **61**, 65–170 (2018)
9. Geis, M.L., Zwicky, A.M.: On invited inferences. Linguist. Inq. **2**(4), 561–566 (1971)
10. Herburger, E.: Conditional perfection: the truth and the whole truth. In: Semantics and Linguistic Theory, vol. 25, pp. 615–635 (2016)
11. Horn, L.R.: From if to iff: conditional perfection as pragmatic strengthening. J. Pragmat. **32**(3), 289–326 (2000)
12. Mayn, A., van Deemter, K.: Evaluating automatic difficulty estimation of logic formalization exercises (2022). arXiv:2204.12197
13. Newstead, S.E.: Conditional reasoning with realistic material. Think. & Reason. **3**(1), 49–76 (1997)

14. Ohm, E., Thompson, V.A.: Everyday reasoning with inducements and advice. Think. & Reason. **10**(3), 241–272 (2004)
15. Oren, N., van Deemter, K., Vasconcelos, W.W.: Argument-based plan explanation. In: Knowledge Engineering Tools and Techniques for AI Planning, pp. 173–188. Springer (2020)
16. Reiter, E., Dale, R.: Building Natural Language Generation Systems. Studies in Natural Language Processing. Cambridge University Press (2000). https://doi.org/10.1017/CBO9780511519857
17. Sevenants, A.: Conditionals and truth table tasks the relevance of irrelevant (2008)
18. Van Canegem-Ardijns, I., Van Belle, W.: Conditionals and types of conditional perfection. J. Pragmat. **40**(2), 349–376 (2008)
19. Van Rooij, R., Franke, M.: Promises and threats with conditionals and disjunctions. Discourse and Grammar: From Sentence Types to Lexical Categories, pp. 69–88 (2012)
20. Wang, J.-t.: On computational sentence generation from logical form. In: COLING 1980 Volume 1: The 8th International Conference on Computational Linguistics (1980)
21. Xu, W., Callison-Burch, C., Napoles, C.: Problems in current text simplification research: new data can help. Trans. Assoc. Comput. Linguist. **3**, 283–297 (2015)