A Rule-based Approach for Karmina Generation

Franky
Shanghai Jiao Tong University
800 Dongchuan Rd., Shanghai, 200240, China

Charles University in Prague, Faculty of Mathematics and Physics
Malostranské náměstí 25, Prague 1, 11800, Czech Republic

franky.id@gmail.com

Abstract

We present our work in generating Karmina, an old Malay poetic form for Indonesian language. Karmina is a poem with two lines that consists of a hook (*sampiran*) on the first line and a message on the second line. One of the unique aspects of Karmina is in the absence of discourse relation between its hook and message. We approached the problem by generating the hooks and the messages in separate processes using predefined schemas and a manually built knowledge base. The Karminas were produced by randomly pairing the messages with the hooks, subject to the constraints imposed on the rhymes and on the structure similarity. Syllabifications were performed on the cue words of the hooks and messages to ensure the generated pairs have matching rhymes. We were able to generate a number of positive examples while still leaving room for improvement, particularly in the generation of the messages, which currently are still limited, and in filtering the negative results.

1 Introduction

Computational creativity is an interesting area of research, since it deals with how a machine can actually produce something new and creative. Creative, in the sense that it is something that usually comes from human's imagination, which is quite abstract, and unexpected from a machine. In this work, we investigated the matter of creativity in language. In particular, we focused our work in the generation of Karmina, an old Malay poetic form for Indonesian language.

Karmina is a poem that consists of two lines with around 8-12 syllables on each line. The first line is called the hook, which acts as the opening line of the poem. The second line is called the message, which contains the meaning of the poem. The language used in Karmina is usually less formal, i.e. closer to conversational language. Karmina resembles *Pantun,* a more well-known form of Malay poetic form, but is different from Pantun in the number of lines it has. It can probably be compared to a couplet in English, in terms of the number of lines and rhymes it must follows. Due to its short presentation, Karmina is also called a *quick* Pantun.

One of the unique aspects of Karmina is in its hook and message relationship. The hook on the first line has no discourse relation with the message on the second line. Take as an example Karmina presented below in Indonesian:

*Gendang gend**ut** tali kecap**i***
*Kenyang per**ut** senanglah hat**i***
(Fat drum string of lute
Full stomach makes a happy heart)

and also our attempt to make one in English:

*Soft meat**ball** is easy to **chew***
*Love them **all** but trust a **few***

The hook in Karmina acts as the entrance of the poem and is used to engage interest or curiosity of the audience. It usually talks about something common in daily life, some unusual or less meaningful information, or obvious facts, e.g. *Buah pisang warnanya kuning* (Banana is yellow).

The message of Karmina contains the real meaning that wants to be delivered by the author. It might contain ideas, jokes, mockeries, or even advices. The sentence used in the message does not need to be formal. It creates its poetic form by omitting some function words, changing the word order, or by using the base form of the word instead of using a morphologically derived one.

The rhyme scheme in Karmina is formally deemed to be *a-a*. But we found that most of the Karminas have the rhyme schemes of (*a b*)-(*a b*), with (*a b)* in the same line, as shown in the examples above. The position of the rhyme *a* in an (*a b*) line is usually located in the middle of the sentence and is determined by how to read the Karmina so that the rhymes on both lines match to each other.

We chose to work with Karmina due to its simple and short presentation. It will be both challenging and interesting to answer the question of whether we can computationally generate a simple and short poem that contains a single idea (meaning), while at the same time maintain its poetic characteristics. From a cultural point of view, we considered this as one of the ways to conserve this poem, as well as to introduce it to others.

We centered our work in generating Karmina with rhyme schemes of (*a b*)-(*a b*). We considered Karmina in this form to be more poetic and have more interesting structure. We present our work by first mentioning some related works in the area of poetry generation in Section 2. In Section 3, we describe our approach for syllabification, hook and message generation, and the construction of the final Karmina. We present the results of our experiments in Section 4. The discussions of findings, issues, and future works are presented in Section 5.

2 Related Works

Some recent works in poetry generation are in the area of English Haiku generation. In Netzer et al. (2009), the authors use word association information to enrich user supplied seed words with their associated words. Candidate lines are produced from pre-extracted lines that match the seed words and their associated words, as well as the chosen syntactic patterns. The poems are generated by random line matching processes and by filtering the generated Haikus based on some constraints and internal associativity scores. The work by Wong and Chun (2008) uses a different approach. They represent the extracted line as a vector of words. The Haikus are produced by generating sentence pairs based on the selected seed words. They are then ranked based on the similarity scores of their lines.

Other previous work in a more general area of poetry is by Manurung et al. (2000). In this work, they proposed a stochastic hill climbing search model that iteratively produces set of candidate solutions and evaluates them based on some defined phonetic, linguistic, and semantic measures. Gervas (2001) and Díaz-agudo et al. (2002) focus their works in the area of Spanish poetry generation. They use prose description and rough specification from user as their input. The appropriate strophic forms are selected based on this input. The process continues using a case based reasoning approach to produce the final poem.

We consider our work to have different focus and pose different challenges. The first thing is that the meaning of a Karmina can be understood directly. This property might be different from other type of poetry which requires deeper interpretation. Hence, the problem usually lies in generating a poem with a deep embedded meaning. The second one is related to the property of the hook that should contain less important information (ignorable) compared to the message. We believe that we could fulfill these two requirements by defining proper schemas and constraints, and by controlling the words used. The last thing to consider is about the absence of discourse relation between the hook and the message. Our current approach to the problem is by generating the hooks and the messages separately using different knowledge base and different constraints. By this treatment, we expect the generated hooks and messages to be independent of each other.

3 Our Current Approach

In his thesis, Manurung (2003) defines three properties that a poem should fulfill: meaningfulness, grammaticality, and poeticness. We think that these three are inherent properties of Karmina and unquestionably should be fulfilled by the generated poem. Meaningfulness is handled by putting constraints on the proposed schemas which restrict the words used in the poem. It is also supported by ensuring the grammaticality of the poem, which is handled by positioning the words inside the schema properly. In terms of poeticness, we consider that Karmina obtains its poeticness through its rhyme structure, limitation on the number of words or syllables, and the forms of the words used. Hence, poeticness is handled by considering these three aspects in the generation of the poem.

We will start this section with the description of the schemas used in generating the hooks and the messages. We will then continue with the explanation of the syllabification algorithm and the generation of the Karmina.

3.1 Generating the Hook

The hook of Karmina can be recognized from its characteristic of somehow sounds like an 'unimportant' utterance, e.g. *kelapa diparut enak rasanya* (*grated coconut tastes good*) or *ikan lele beli di pasar* (*catfish bought from the market*). In our first attempt, we took text segments from the collections of news, blogs, and reviews websites. The segments were produced by splitting the sentences using punctuations, such as comma, period, question mark, single and double quotes, and exclamation mark. We were hoping to find segments that could be used as hooks. But, we found that this kind of utterance is quite rare.

We looked deeper into some of the examples of Karmina and found something interesting. The majority of the hooks that we met have some similar syntactic and semantic patterns. We analyzed the examples and came out with a set of schemas to generate the hooks. One property of Karmina that we think makes the generation of the hook possible is that a sentence in Karmina usually consists of only 4-5 words. We defined around 19 schemas for the hook. Some of them differ only in their word order, e.g. a sentence with a word order of `X Y Z` and a sentence with a word order of `X Z Y`, where `X`, `Y`, `Z` can be noun, verb, adjective, etc. These schemas are not exhaustive. They cover some of the hooks that we found on our small examples. Other forms of hooks may also present.

The knowledge base was built manually by finding all suitable nouns, verbs, and other necessary information. We did some categorization on them, e.g. as fish, flower, tree, location, and specified their relations as required in the schemas. We describe in this section the first three schemas that we defined. We use "," (comma) to denote a conjunction and ";" (semicolon) to denote a disjunction.

Schema 1

Dahulu X *sekarang* Y

Constraints

```
Noun(X), Noun(Y), ChangeTo(X,Y),
Length(X,1), Length (Y,1)
```

In **Schema 1**, the generated hook will have a meaning of *before* (*dahulu*) and *after/now* (*sekarang*). In this case, `X` and `Y` are usually replaced by nouns that have this kind of relationship. The replacement using other word classes is also possible. We restricted `X` and `Y` to noun since it is the most common class we saw on the examples. In order to check for this relationship, we defined a predicate `ChangeTo` that check for two things from the knowledge base:

- Whether `X` can be made from `Y` and vice versa, e.g. knife is made from iron.
- Whether `X` is better than `Y` and vice versa, e.g. gold is better than silver.

Predicate `Length` checks for how many words the noun `X` and `Y` has, which we limit to 1, to maintain the poeticness of the generated hook. In our current work, we used the number of words instead of syllables to simplify the word selection process, with the assumption that the number of syllables inside a word is around two to four syllables.

Schema 2

Sudah X Y *pula*

Constraints

```
Noun(X), Noun(Y), SameType(X,Y),
Length(X,1), Length(Y,1), (Tree(X);
Flower(X); Food(X))
```

Schema 2 was made from one of the examples that we found. The `X` and `Y` come from the same category, i.e. both are the name of fish, bird, vegetable, island, tree, etc. The meaning of the generated hook will be that `Y` is redundant because `X` is already present. We restricted `X` and `Y` to be in the same category to give an emphasis on this redundancy. We used tree, flower, and food for the categories. This was based on our experiments that using other categories resulted in a sentence with odd meaning.

```
Schema 3
  X Propnya Y
Constraints
  Noun(X), Adjective(Y), Has(X, Prop,
  Y) Length(X,2), Length(Y,1)
```

In **Schema 3**, the generated hook simply means X with a property Prop that has the value of Y. For example, X can be a banana (*buah pisang*) with a property of color (*warna*) and property's value of yellow (*kuning*). Hence, the generated hook will be *Buah pisang warnanya kuning* (Banana has a yellow color). We found that this kind of hook is quite often used.

3.2 Generating the Message

The message of Karmina is more free in its meaning and structure. Creating all possible schemas is not a feasible option. However, we managed to find messages that follow certain schemas. They have the same structures with the **Schema 1** and **Schema 2**. Hence, in this work, the message was generated by using these two schemas only. These two schemas bind the hook and the message to have the same structure, i.e. both have the structures of *Dahulu* X *sekarang* Y or *Sudah* X Y *pula*. They differ in the types and constraints of the X and Y used. We experimented using a list of positive and negative sentiment words to replace X and Y.

For **Schema 1**, X and Y were replaced by words that have different sentiment (positive-negative or negative-positive). These two words are antonym to each other. The generated message will have a meaning of a change from a positive (good) to negative (bad) condition, or vice versa, e.g. *Dahulu kaya sekarang miskin* (was rich but now poor).

In **Schema 2**, X and Y were replaced by words that have the same sentiment. We expected the resulting sentences to contain the repetition of two good or two bad expressions and hence, intensifying the positive or negative condition. For example, *Sudah busuk bau pula* (rotten and stink).

To our knowledge, there are no subjectivity lexicons for Indonesian. Hence, we produced the list by translating English subjectivity lexicon (Hu and Liu, 2004), which originally has 2006 positive words and 4783 negative words, using Google Translate. The translation results were then filtered manually to remove untranslated words, bad translations, and words that do not contain positive or negative sentiment. The final lexicon contains 740 positive words and 1500 negative words.

3.3 Syllabification

The syllabification is used in searching for the hooks that rhyme with the messages. We used a set of rules to cut syllables out of the word iteratively. The syllabification starts from the front and by looking into the pattern of the first 3-6 letters of the word. We defined rules for the possible patterns that determine how many letters from the front that will be taken as a syllable. The syllable is cut out from the word and the iteration continues with the truncated word. The iteration stops when only two or less letters are left. The patterns are the combinations of vowel-consonant patterns and alphabet letters. The vowel-consonant pattern is simply a sequence of *v* (vowel) and *c* (consonant) marker. There are only five vowels in Indonesian (*a,i,u,e,o*).

```
kecapi (cvcvcv) (take first 2)
capi   (cvcv)   (take first 2)
pi     (cv)     (take all)
```

Figure 1. Syllabification of *kecapi* to (*ke*, *ca*, *pi*)

The example in Figure 1 shows the word *kecapi* matches the cvcv pattern, and the rule specifies to take the first two letters from the front (ke) as a syllable. The truncated word *capi* also falls into the same rule. In the last step, only two letters are left and we took them all as a syllable.

3.4 Rhymes

In our work, we used two types of rhymes, perfect and imperfect (half). In Indonesian, the pronunciation of a word can be determined from its syllables and hence, we can check whether two words rhyme with each other by matching their last syllables. For perfect rhyme, we considered two words as having perfect rhyme if they have the same last syllables. For imperfect rhyme, we divided the case into two. If the last letter of the last syllable is a vowel, we took this vowel to be compared. If the last letter is a consonant, we searched for the first vowel from the last after the consonant and took the vowel together with the following consonants to be compared. For diphthong (*ai, au, oi*), we took both of the vowels to be compared.

3.5 Constructing the Karmina

The Karmina was produced by first generating a list of hooks and a list of messages. The generation processes were done separately for the hook and the message. We selected one of the messages and we tried to find a proper hook for the message from the list of hooks.

Syllabifications were performed on the cue words of the selected message and on the cue words of the hooks in the list. Cue words of hook or message are the middle word and the last word of the sentence. Given the schema, we can usually determine the second word as the middle word.

We produced a list of possible hooks for the selected message by selecting hooks that rhyme with the message, producing *(a b)-(a b)* rhyme scheme. It was done by comparing the last syllables of the cue words of the message and the hooks. We differentiated the hooks which have perfect rhymes with the message and the hooks which have imperfect rhymes. We gave higher priority to the hooks that rhyme perfectly with the message. If no such hooks exist, we took the hook from the later.

Message generated using **`Schema 1`** or **`Schema 2`** could only take the hook that has the same structure. Hence, in this work, the generated Karmina could only have the structure of **`Schema 1`** or **`Schema 2`** on both of its lines.

The final Karmina was produced by pairing the selected message with one of the possible hooks which was selected randomly from the list.

4 Experiments

We implemented our work for syllabification and Karmina construction in Perl, and generation of the hooks and the messages in Prolog. We evaluated the syllabification on a small list of 258 unique words taken from two news articles. We found that 16 words were incorrectly syllabified. The main causes are due to incomplete rules, foreign words or abbreviations, and ambiguous words. Examples of ambiguous words are *beribu* that can be read as *ber-ibu* or *be-ribu*, and words that contain diphthong-like string such as *b**ai**kan* and *bag**ai**kan*. In the first word, the *ai* is not a diphthong. Both cases might require context disambiguation and lemmatization which are not covered in the current rule-based syllabification.

For Karmina evaluation, first we generated lists of all possible hooks and messages from **`Schema 1`** and **`Schema 2`**. Next, we generated all possible Karminas from these lists. However, we found that all generated Karminas were in the form of **`Schema 2`**. We failed to generate Karmina for **`Schema 1`** since there were no hooks and messages that rhyme with each other due to small number of hooks and messages that we have for **`Schema 1`**. Table 1 shows the evaluation results.

Table 1. Karmina Evaluation

Hook	Message	Total
Proper	Proper	10
Proper	Not Proper	30
Not Proper	Proper	1
Not Proper	Not Proper	59
		100

The evaluation was performed on 100 randomly selected Karminas. The proper and improper annotations were done through discussions by two native speakers. We managed to get 10 Karminas with acceptable hooks and messages. We found that the improper hook was mainly caused by the use of uncommon names e.g. *holdi*, *hamboi*. The other cause was that **`X`** and **`Y`** in **`Schema 2`** may sometimes not be able to be placed side by side, e.g. *Sudah tomat srikaya pula* (tomato and sugar-apple*)*. Although both of the objects are fruits, the more common perception of tomato is as vegetables and hence, the sentence sounds strange.

a)	*Sudah leci menteng pula*	(lychees and menteng)
	Sudah ahli tampan pula	(skilled and handsome)
b)	*Sudah kiwi ceri pula*	(cherry and kiwi)
	Sudah ahli kejujuran pula	(skilled and honesty)

Figure 2. (a) Positive example (b) Negative example

For the message, the main cause was as shown in Figure 2 (b). The sentence sounds unusual because it combines adjective *skilled* with noun *honesty*. This happened because of the incomplete constraints on the schema, i.e. no restriction in the part of speech of `X` and `Y`. Other reason was because of words that do not fit to be put together, e.g. *Sudah agung bagus pula* (majestic and smart).

5 Discussion and Future Works

In this section, we discuss several findings and issues that we found, and our future plans for the work.

Incomplete Constraints. The constraints in the schema are the most crucial parts of the generation process. The grammaticality, meaning, and poeticness of the generated sentence depend on the constraints used. Hence, some of the problems as the one shown in Figure 2 were caused by incomplete specification of the constraints.

Manual Intervention. The main issue in utilizing knowledge base and schemas is in the amount of manual work that needs to be performed in creating them. One of the problems is in the information collection, such as collecting property of nouns (e.g. skin as property of a fish and the skin of a fish is slippery), antonyms, and what kind of verb that can match certain noun (e.g. coconut can be grated). Currently, the information was built manually and hence slowly.

One of the options to automate the knowledge creation such as 'what *object* has what kind of *property* with what *value*' is by first generating all possible combinations of nouns and properties, and finding a way to validate this knowledge efficiently. Data that contains this information might be needed for validation. Another option might be to query a search engine. Using collocation information in the results, we may somehow validate the knowledge. However, the policy of automated query of current search engine might be a hurdle.

The other issue is if we want to automate the creation of the schema. Using extracted sentences and part of speech information may be useful. But this approach might not be enough, since we also need to capture the dependency between items. Without deeper constraints, the extracted schema will be just a shallow representation.

Filtering the Knowledge. We found that having too much knowledge about certain things can actually result in a less or non-poetic sentence, e.g. a hook *kod pasifik kulitnya licin* (pacific cod has slippery skin). Grammatically and semantically, there is nothing wrong with the sentence. The problem lies in the use of *kod pasifik* (pacific cod) that is rarely mentioned in a normal daily life. Since the hook is usually about something common to the majority of the audience, using a rare term like this might cause it to lose its poeticness.

Corpus Based Approach for Message Generation. We are considering the corpus-based approach that utilizes the segments extracted from the corpus for message generation. Contrary to the hook, we found that the use of segments for the message is more promising. We experimented with blog corpus, since we considered it as the most proper corpus, because of its informal and conversational style language. We picked segments that have length (number of words) greater than two, and for poeticness reason, do not start with certain function words. The chosen segments were further processed by normalizing slang words, e.g. *gw* to *saya*. Further removal of unpoetic function words (*yang, adalah, untuk*) was performed. The final segments that have length more than three were stored. The Karmina generation was performed using the same procedure. We determined the middle word of the message by taking the word where the fourth syllable is located. Figure 3 shows the positive examples that we were able to generate. One important aspect that we still need to consider is about the characteristics of the segments that can be considered as good messages.

Ikan cakalang di danau emas	(tuna in lake emas)
Selamat ulang tahun ya mas	(happy birthday)
Sungai bengkulu sungai bilah	(bengkulu and bilah river)
Aku malu kepada allah	(i am ashamed of god)

Figure 3. Positive examples of Karmina using corpus based approach for message part

6 Conclusions

We described our work in Karmina generation that utilized a rule-based approach in generating the hooks and the messages. We considered the notion of grammaticality, meaningfulness, and poeticness by defining proper schemas and constraints. We also discussed some of the problems and future improvements in section 5. We concluded that the rule-based approach is able to produce some positive examples. Some limitations still exist, especially in the message generation, and a lot of improvements are still needed to produce more of proper Karmina. We are considering the corpus-based approach in our future work for the message generation and a more automated approach in knowledge collection.

References

B. Díaz-agudo , P. Gervás, and P.A. González-calero. 2002. Poetry Generation in COLIBRI. *Advances in Case-Based Reasoning* (2002): 157-159.

H.M. Manurung, G. Ritchie, and H. Thompson. 2000. Towards a computational model of poetry generation.In *Proc. of the AISB'00*.

H.M. Manurung. 2003. An evolutionary algorithm approach to poetry generation. Ph.D. thesis, University of Edinburgh.

J. M. Toivanen, H.Toivonen, A.Valitutti, and O. Gross 2012. Corpus-based generation of content and form in poetry. In *International Conference on Computational Creativity, Dublin, Ireland* (pp. 175-179).

M. Hu and B. Liu. 2004. Mining and Summarizing Customer Reviews. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2004)*.

M. Tsan Wong and A. Hon Wai Chun. 2008. Automatic Haiku generation using VSM. In *Proc. of ACACOS'08*, April.

N. Tosa, H. Obara, and M. Minoh. 2008. Hitch haiku: An interactive supporting system for composing haiku poem. In *Proc. of the 7th International Conference on Entertainment Computing*.

P. Gervas. 2001. An expert system for the composition of formal Spanish poetry. *Journal of Knowledge-Based Systems*, 14.

P. Gervas. 2002. Exploring quantitative evaluations of the creativity of automatic poets. In *Proc. of the 2nd Workshop on Creative Systems, Approaches to Creativity in Artificial Intelligence and Cognitive Science, the 15th European Conf. on Artificial Intelligence (ECAI 2002)*.

Y. Netzer, D.Gabay, Y.Goldberg, and M.Elhadad. 2009. Gaiku: generating Haiku with word associations norms. In *Proceedings of the Workshop on Computational Approaches to Linguistic Creativity*.

Appendix A. Schemas for Hook Generation

The rest of the schemas used for generating the hooks are provided below as references. Some additional relations that might need to be explained are:

- `Location(X)`: `X` is a location, e.g. name of a mountain, name of a river, etc.
- `LocationType(X)`: location type of `X`, e.g. `X` is a river, a mountain, or other abstract types such as on the top of an object (*on a table*), inside an object, etc.
- `Has(X,Y,A,B)`: `Y` is performed on `X`, resulting in `X` with property `A` and property value of `B`, e.g. `Has(fish, fried, taste, good)`.

We use Prolog notations such as "," (comma) to denote a conjunction, ";" (semicolon) to denote a disjunction, and "_" (underscore) to denote any matching term.

Schema 4

Sudah ke X *ke* Y *pula*

Constraints

```
Location(X), Location(Y), Location-
Type(X)==LocationType(Y),
Length(X,1), Length (Y,1)
```

Schema 5

X Y Prop*nya*

Constraints

```
Noun(X), Adjective(Y), Has(X,Prop,Y)
Length(X,2), Length(Y,1)
```

Schema 6

X *di*Z Y Prop*nya*

Constraints

```
Noun(X), Verb(Z), Adjective(Y),
Has(X,Z,Prop,Y)
```

Schema 7

X *di*Z Prop*nya* Y

Constraints

```
Noun(X), Verb(Z), Adjective(Y),
Has(X,Z,Prop,Y)
```

Schema 8

X Z *di* Y

Constraints

```
Noun(X), Verb(Z), Location(Y),
not(Location(X)), Has(X,Z,_,_),
Length(X,1)
```

Schema 9

X *di* Y

Constraints

```
Noun(X), Location(Y),
not(Location(X)), Length(X,2),
Length(Y,2)
```

Schema 10

```
X Y
```

Constraints

```
Noun(X), Noun(Y), SameType(X,Y),
Length(X,2), Length(Y,2)
```

Schema 11

```
X X Y Y
```

Constraints

```
Noun(X), Noun(Y), SameType(X,Y),
Length(X,1), Length(Y,1)
```

Schema 12

```
X Y Y
```

Constraints

```
Noun(X), Noun(Y), SameType(X,Y),
Length(X,2), Length(Y,1)
```

Schema 13

```
X X Y
```

Constraints

```
Noun(X), Noun(Y), SameType(X,Y),
Length(X,1), Length(Y,2)
```

Schema 14

```
X Y A A
```

Constraints

```
Noun(X), Adjective(Y), Noun(A),
Has(X,_,Y), SameType(X,A),
Length(X,1), Length(A,1)
```

Schema 15

```
X Y A B
```

Constraints

```
Noun(X), Adjective(Y), Noun(A), Ad-
jective(B), Has(X,_,Y), Has(A,_,B),
Length(X,1), Length(A,1)
```

Schema 16

```
X Y A
```

Constraints

```
Noun(X), Adjective(Y), Noun(A),
Has(X,_,Y), SameType(X,A),
Length(X,1), Length(A,2)
```

Schema 17

```
X A B
```

Constraints

```
Noun(X), Noun(A), Adjective(B),
Has(A,_,B), SameType(X,A),
Length(X,2), Length(A,1)
```

Schema 18

X Z Y Prop*nya*

Constraints

```
Noun(X), Verb(Z), Adjective(Y),
Has(X,Z,Prop,Y)
```

Schema 19

X Z Prop*nya* Y

Constraints

```
Noun(X), Verb(Z), Adjective(Y),
Has(X,Z,Prop,Y)
```