WISeR Guidelines

WISeR is an annotation scheme that represents the semantics of an utterance. It attempts to revise the current AMR guidelines to be more interpretable by parsers. AMR makes use of PropBank frames which encode thematic roles using numbered arguments (`ARGx`).

`ARG0`	agent
`ARG1`	patient
`ARG2`	instrument, benefactive, attribute
`ARG3`	starting point, benefactive, attribute
`ARG4`	ending point
`ARGM`	modifier

Table 1: List of arguments in PropBank

Crucially, while thematic roles are meaning relations, numbered `ARG`s only indirectly encode meaning. In practice, there can be a disconnect between numbered `ARG`s and the thematic roles they are meant to encode. Notice that the benefactive and attribute roles can be encoded by both `ARG2` and `ARG3`. As a result, numbered `ARG`s are often semantically overloaded.

Additionally, many numbered `ARG`s are assigned the fine-grained role 'verb specific' which does not form a meaningful class and thus may cause problems for parsers and annotators.

WISeR does not use numbered `ARG`s. Instead it encodes thematic role directly as WISeR relations. Below we give a comprehensive list of meaning relations available to WISeR 1.0. For each role, we will give some simple illustrative examples as well as some harder edge cases. We believe that by providing many different examples for each case, annotators will be able to use these guidelines as a reference manual when he or she is uncertain about choosing an appropriate representation. We have also included an index which includes traditional thematic roles and naive descriptions of relations to help annotators look up the appropriate WISeR relation. The relations are given roughly in order of frequency and are grouped together with other similar meaning relations. WISeR adopts non-core roles from AMR and uses them in place of numbered `ARG2-6` and `ARGM`. WISeR also introduces a small number of new roles based on VerbNet roles.

Note to annotators

The WISeR annotation scheme shares many similarities with AMR. The primary differences are: (i) disposing of sense relations, and (ii) replacing numbered `ARG`s with WISeR relations.

If you encounter something which is not covered here, it will likely be covered in the AMR guidelines: https://github.com/amrisi/amr-guidelines/blob/master/amr.md

1 Core roles

1.1 Actor

WISeR makes use of the `actor` relation to encompass the traditional thematic role of agent.

The boy wants the girl to believe him

```
(w / want
   :actor (b / boy)
   :theme (b2 / believe
              :actor (g / girl)
              :theme b))
```

However, the `actor` relation is less specific than a thematic agent. An agent must be intentional, while the `actor` relation may also include non-intentional doers. The `actor` role corresponds to the thing which is the impetus behind the event.

The bus hit the curb

```
(h / hit
   :actor (b / bus)
   :theme (c / curb))
```

The role `actor` is also used to annotate the subject of a communication verb.

The boy said that the bus crashed

```
(s / say
   :actor (b / boy)
   :theme (c / crash
             :theme (b / bus)))
```

Importantly, there is no one-to-one correspondence between the role of `actor` and the notion of grammatical subject. Firstly, a subject is not always an actor (See also `theme` §1.2 and `benefactive` §1.3).

Secondly, there are `actor` arguments which are not always grammatical subjects. For instance, WISeR (following PropBank) treats the entity or event which instils an emotion in a `theme` to be an `actor`.

The boy is scared of the monkey
The monkey scares the boy

```
(s / scare
   :actor (m / monkey)
   :theme (b / boy))
```

Even when there is no transitive verbal form of the predicate (e.g., *afraid*) the `actor` is still the entity which instils the emotion in the `theme`. In the following sentence, the monkey is the impetus of the fear.

The boy is afraid of the monkey

```
(a / afraid
   :actor (m / monkey)
   :theme (b / boy))
```

As mentioned, emotive predicates may even have an eventive `actor`.

The boy is glad that the monkey left

```
(g / glad
   :actor (l / leave
             :actor (m / monkey))
   :theme (b / boy))
```

Finally, the subject of perception predicates (e.g., *see* and *hear*) is treated as an `actor` because it is doing the perceiving, even if unintentionally.[1]

The boy saw the horse in the garden

```
(s / see
   :actor (b / boy)
   :theme (h / horse
             :location (g / garden)))
```

1.2 Theme

WISeR does not distinguish between the thematic roles patient and theme. The role `theme` is used for arguments which either undergo an action or have some property.

The boy hugged the monkey

```
(h / hug
   :actor (b / boy)
   :theme (m / monkey))
```

A `theme` may also appear as the grammatical subject. For instance, in an unaccusative construction.

The vase broke

```
(b / break
   :theme (v / vase))
```

This retains its role in a causative construction when it occurs as the direct object and the `actor` is added as the grammatical subject.

The wind broke the vase

```
(b / break
   :actor (w / wind)
   :theme (v/ vase))
```

A less obvious case of a `theme` is the subject of a verb like intransitive *roll*.

[1] Those who are familiar with thematic roles might notice that we annotate thematic experiencers sometimes as a `theme` (as with emotive predicates like *afraid*) and sometimes as an `actor` (as with verbs of perception like *see*). Likewise, we sometimes annotate the so-called thematic stimulus as an `actor` (emotive predicates) and sometimes as a `theme` (verbs of perception). This is in keeping with PropBank, and we agree that it is the most natural way to annotate these constructions without introducing more relations.

The boy rolled down the hill

```
(r / roll
   :theme (b / boy)
   :direction (d / down)
   :path (h / hill))
```

Compare this to the following.

The girl rolled the boy down the hill

```
(r / roll
   :actor (g / girl)
   :theme (b / boy)
   :direction (d / down)
   :path (h / hill))
```

If it is clear from the context that the boy is the impetus behind the event, then the annotator can ascribe the concept `boy` both thematic relations.

The boy rolled down the hill on purpose

```
(r / roll
   :actor (b / boy)
   :theme b
   :direction (d / down)
   :path (h / hill)
   :manner (o / on-purpose))
```

An even more striking example is the verb *drive*.

The car drove west

```
(d / drive
   :theme (c / car)
   :direction (w / west))
```

The girl drove the car west

```
(d / drive
   :actor (g / girl)
   :theme (c / car)
   :direction (w / west))
```

As a rule of thumb, if the subject of an intransitive verb can also appear as the object when the verb is transitive, it is likely a `theme`.

The role `theme` is also used to annotate the message communicated by a communication verb.

The boy said that the bus crashed

```
(s / say
   :actor (b / boy)
   :theme (c / crash
             :theme (b / bus)))
```

As well as propositions which are embedded under a modal concept.[2]

The boy can ski
It is possible the boy is skiing
The boy might ski

```
(p / possible
   :theme (s / ski
             :actor (b / boy)))
```

The boy must clean the house
The boy is obligated to clean the house
It's obligatory that the boy clean the house

```
(o / obligate
   :theme (c / clean
             :actor (b / boy)
             :theme (h / house)))
```

The theme relation is also used when an argument has the property described by the predicate.

The girl is tall

```
(t / tall
   :theme (g / girl))
```

The boy is glad

```
(g / glad
   :theme (b / boy))
```

1.3 Benefactive

The `benefactive` role is used when representing a number of constructions. Most notably, it is used to represent a recipient in a dative or double object construction.

The girl gave a book to her friend
The girl gave her friend a book

```
(g / give
   :actor (g2 / girl)
   :theme (b / book)
   :benefactive (f / friend
                   :poss g2)))
```

It is also used for some (but not all) other arguments introduced by prepositions such as *to* and *for* (See also `asset` §1.4 and `purpose` §4.2).

[2] More discussion of modality is included in the AMR guidelines.

The girl sings to her cat
The girl sings for her cat

```
(s / sing
   :actor (g / girl)
   :benefactive (c / cat))
```

The role `benefactive` is used when the argument is either a recipient or an individual/organisation for whose benefit or detriment an action is done (i.e., they are benefited or harmed by the event).

The dice fell kindly for the girl

```
(f / fall
   :theme (d / dice)
   :manner (k / kind)
   :benefactive (g / girl))
```

The role `benefactive` is also used to annotate the addressee or hearer of a communication verb.

The boy said to the girl that the bus crashed

```
(s / say
   :actor (b / boy)
   :benefactive (g / girl)
   :theme (c / crash
             :theme (b / bus)))
```

The boy told the girl that it was raining

```
(t / tell
   :actor (b / boy)
   :benefactive (g / girl)
   :theme (r / rain))
```

The boy ordered the girl to clean her room

```
(o / order
   :actor (b / boy)
   :benefactive (g / girl)
   :theme (c / clean
             :actor g
             :theme (r / room
                       :poss g)))
```

As well as arguments of permission and obligation modals.

The girl permitted the boy to eat a cookie

```
(p / permit
   :actor (g / girl
   :benefactive (b / boy)
   :theme (e / eat
             :actor b
             :theme (c / cookie)))
```

The girl obligated the boy to clean the house

```
(o / obligate
   :actor (g / girl
   :benefactive (b / boy)
   :theme (c / clean
             :actor b
             :theme (h / house)))
```

Notice that a `benefactive` role should not be confused with the notion of a beneficiary as the `benefactive` argument may be negatively affected.

The girl laid a trap for the monkey

```
(l / lay
   :actor (g / girl)
   :theme (t / trap)
   :benefactive (m / monkey))
```

Some verbs (e.g., *receive*) seem like they should have a `benficative` subject. However, for the sake of maintaining consistency with both PropBank and VerbNet, we assign this subject the `actor` role.

The girl received a fine

```
(r / receive
   :actor (g / girl)
   :theme (g2 / gift))
```

1.4 Asset

In a bid to reduce verb specific arguments, WISeR makes use of the VerbNet relation `asset`. This is used with predicates which describe exchanges and transactions such as `buy`, `sell`, `offer`, `order`, as well as many others. We use the `asset` relation for any argument which moves in the opposite direction to the `theme` in an exchange.

The girl bought the axe for twenty dollars with a credit card
The girl bought the axe with a credit card for twenty dollars

```
(b / buy
   :actor (g / girl)
   :theme (a / axe)
   :asset (m / monetary-quantity
             :quant 20
             :unit (d / dollar)))
   :instrument (c / card
                  :mod (c2 / credit)))
```

Unlike the VerbNet role, the WISeR role `asset` is not restricted to monetary prices. For instance, verbs like `refund` or `rebate` have a `theme` which is typically a `monetary-quantity` while the `asset` is the thing exchanged in order to receive the money.

The cashier refunded the girl twenty dollars for the axe
The cashier refunded twenty dollars to the girl for the axe

```
(r / refund
   :actor (c / cashier)
   :benefactive (g / girl)
   :theme (m / monetary-quantity
             :quant 20
             :unit (d / dollar))
   :asset (a / axe))
```

1.5 Instrument

The `instrument` relation is used for arguments which describe a thing used in carrying out an action. In English, `instruments` are usually introduced by the preposition *with*.

The girl chopped the wood with the axe

```
(c / chop
   :actor (g / girl)
   :theme (w / wood)
   :instrument (a / axe))
```

Notice that, in the following example, `axe` is not an `instrument` of `chop` despite being used in the chopping event because it is not an argument of `chop`.

The girl used the axe to chop the wood

```
(u / use
   :actor (g / girl)
   :theme (a / axe)
   :purpose (c / chop
               :actor g
               :theme (w / wood)))
```

1.6 Topic

The `topic` relation is used to annotate the subject-matter of an entity or event.

The professor wrote about math

```
(w / write
   :actor (p / professor)
   :topic (m / math))
```

The professor of math taught at the university
The math professor taught at the university

```
(t / teach
   :actor (p / professor
             :topic (m / math))
   :location (u / university))
```

The employee emailed the president of human resources

```
(e / email
   :actor (e2 / employee)
   :theme (p / president
             :topic (r / resources
                       :mod (h / human))))
```

The problem with deregulation

```
(p / problem
   :topic (d / deregulation))
```

1.7 Manner

The `manner` relation is used for arguments which describe the way something happens. It provides an answer to the question: "how was it done?". Notice that, when using `manner`, we drop the *-ly* from the end of the adverb.

The boy sang beautifully

```
(s / sing
   :actor (b / boy)
   :manner (b2 / beautiful))
```

It can also represent the means by which something is done.

The boss decreased spending by shortening hours

```
(d / decrease
   :actor (b / boss)
   :theme (s / spending)
   :manner (s2 / shorten
              :actor b
              :theme (h / hours)))
```

WISeR also uses `manner` to represent arguments which AMR handles with a `medium` role. Note that this use of `manner` differs from `instrument`, as `instrument` relations describe the thing used, whereas this use of `manner` describes a more general means.

The girl talked in French

```
(t / talk
   :actor (g / girl)
   :manner (l / language :wiki "French_language"
              :name (n / name :op1 "French")))
```

The boy told the girl by email

```
(t / tell
   :actor (b / boy)
   :theme (g / girl)
   :manner (e / email))
```

1.8 Accompanier

The `accompanier` relation is used for arguments that accompany another in the event.

The nanny walked to town with the newborn

```
(w / walk
   :actor (n / nanny)
   :end (t / town)
   :accompanier (n / newborn))
```

This differs from an `actor` as the `accompanier` may not able to perform the event on its own.

The boy went to school with his backpack

```
(g / go
   :actor (b / boy)
   :end (s / school)
   :accompanier (b2 / backpack
                    :poss b))
```

2 Spatial

2.1 Location

The role `location` is used to represent constituents which describe where an event took place.

The man died in his house

```
(d / die
   :theme (m / man)
   :location (h / house
                :poss m))
```

The man died near his house

```
(d / die
   :theme (m / man)
   :location (n / near
                :op1 (h / house
                        :poss m)))
```

The man died between his house and the river

```
(d / die
   :theme (m / man)
   :location (b / between
                :op1 (h / house
                        :poss m)
                :op2 (r / river)))
```

The detective arrived at the scene of the crime

```
(a / arrive
   :theme (d / detective)
   :end (s / scene
           :location-of (c / crime)))
```

The `location` role can also be used for some verbal arguments.

The man fit three marshmallows in his mouth

```
(f / fit
   :actor (m / man)
   :theme (m2 / marshmallow
              :quant 3)
   :location (m3 / mouth
                 :part-of m))
```

2.2 Direction and Path

The relations `direction` and `path` can represent arguments and modifiers of verbs of movement. Either one of these relations may be present without the other. In the following example we know the `path` of the bouncing, but not the direction.

The ball bounced along the street

```
(b / bounce
   :theme (b / ball)
   :path (a / along
            :op1 (s / street)))
```

Similarly, we might know the `direction` but not the `path`.

The car drove west

```
(d / drive
   :theme (c / car)
   :direction (w / west))
```

Besides the cardinal directions (*north, south, east, west*), other typical directions include *up, down, back, left, right, through, over*, etc.[3]

A `direction` may also appear within a `path` argument.

The soldiers marched east along the road to Moscow

```
(m / march
   :actor (s / soldier)
   :direction (e / east)
   :path (a / along
            :op1 (r / road
                    :direction (c / city
                                  :wiki "Moscow"
                                  :name (n / name
                                           :op1 "Moscow")))))
```

2.3 Start and End

The relations `start` and `end` are generally used for changes in location (corresponding to the AMR roles `source` and `destination`) or changes in state. Examples of locative `start` and `end` are given below.

The monkey jumped from tree to tree

```
(j / jump
   :actor (m / monkey)
   :start (t / tree)
   :end (t2 / tree))
```

He drove west, from Houston to Austin

```
(d / drive
   :actor (h / he)
   :direction (w / west)
   :start (c / city :wiki ‘‘Houston’’
             :name (n / name :op1 ‘‘Houston’’))
   :end (c2 / city :wiki ‘‘Austin,_Texas’’
            :name (n2 / name :op1 "Austin")))
```

They can also be used for more abstract directional arguments.

They are descended from royalty

```
(d / descend
   :theme (t / they)
   :start (r / royalty))
```

[3]For the sake of the annotation exercises we will not use the `wiki` role.

WISeR also uses `start` for initial states or materials in verbs of creation (i.e., the `material` role of VerbNet), and `end` for the thing created (i.e., the `product` role of VerbNet).

She cast the bronze into a statue
She cast a statue out of bronze

```
(c / cast
   :actor (h / he)
   :start (b / bronze)
   :end (s / statue))
```

She made a dress out of her curtains
She made her curtains into a dress

```
(m / make
   :actor (s / she)
   :start (c / curtains
              :poss s))
   :end (d / dress))
```

He folded the paper into a card
He folded a card out of the paper

```
(f / fold
   :actor (s / she)
   :start (p / paper)
   :end (c / card))
```

As well as certain verb specific arguments.

The monkey arranged the bananas from a neat stack into a messy pile

```
(a / arrange
   :actor (m / monkey)
   :theme (b / banana)
   :start (s / stack
             :mod (n / neat)
   :end (p / pile
           :mod (m2 / messy))
```

Annotators should also be careful with locative alternations. These involve a `theme` and an `end`.

He sprayed paint onto the wall
He sprayed the wall with paint

```
(s / spray
   :theme (p / paint)
   :end (w / wall))
```

He loaded hay onto the cart
He loaded the cart with hay

```
(l / load
   :theme (h / hay)
   :end (c / cart))
```

However, the `end` can also appear without the `theme`. So annotators should be particularly careful not to assign the `theme` role to the `end` here.

He sprayed the wall

```
(s / spray
   :end (w / wall))
```

He loaded the cart

```
(l / load
   :end (c / cart))
```

An annotator might also wonder whether we could also annotate a `benefactive` as an `end` in a transfer of possession verbs. For instance, in the following example.

The girl gave a dog to the boy

```
(g / give
   :actor (g / girl)
   :theme (d / dog)
   :benefactive (b / boy))
```

WISeR opts to prioritize `benefactive` above `end`. If the argument could best be described as a "recipient", as in this example, you should use `benefactive`.

Finally, it might be hard to tell the difference between an `end` and a `direction`.

The girl threw the pie at the boy

```
(t / throw
   :actor (g / girl)
   :theme (p / pie)
   :end (b / boy))
```

Typically, a `direction` is a word such as *up, down, left, right, north, south, east, west, over, under, through* etc. or a place like a country or city (See `direction` §2.2).

3 Temporal

3.1 Time

The `time` relation establishes when an event took place.

The robbery happened yesterday

```
(r / robbery
   :time (y / yesterday)
```

The bridge was built in December

```
(b / build
   :theme (b2 / bridge)
   :time (d / date-entity
            :month 12))
```

It can also be used for relative time.

The woman had just eaten lunch

```
(e / eat
   :actor (w / woman)
   :theme (l / lunch)
   :time (r / recent))
```

In addition, the `time` relation can equate the time of two events.

The woman frowned when the baby cried

```
(f / frown
   :actor (w / woman)
   :time (c / cry)
            :actor (b / baby)))
```

3.2 Duration

The `duration` relation describes the amount of time over which an event occurs.

He worked for two hours yesterday

```
(w / work
   :actor (h / he)
   :duration (t / temporal-quantity
                :quant 2
                :unit (h2 / hour))
   :time (y / yesterday))
```

The investigator searched for a long time

```
(s / search
   :actor (i / investigator)
   :duration (l / long)
```

The athlete finished the marathon in two hours

```
(f / finish
   :actor (a / athlete)
   :theme (r / run)
             :actor a
             :theme (m / marathon)
             :duration (t / temporal-quantity
                          :quant 2
                          :unit (h / hour)))
```

3.3 Frequency

The `frequency` relation describes how often something occurs.

The phone rang three times

```
(r / ring
   :theme (p / phone)
   :frequency 3)
```

It can also be used to represent quantificational temporal adverbs.

She always eats breakfast

```
(e / eat
   :actor (s / she)
   :theme (b / breakfast)
   :frequency (a / always))
```

3.4 Range

The `range` relation is used to describe a period of time over which an event occurs. This is different from `duration`, because it does not measure the length of the event. Rather, it establishes a period of time in which the event occurs.

His first drink in 3 years

```
(d / drink
   :actor (h / he)
   :ord (o / ordinal-entity
           :value 1
           :range (t / temporal-quantity
                     :  quant 3
                     :  unit (y / year))))
```

Notice in the next example that if we had used the `duration` role, the sentence would mean *"it did not snow for 10 years"*, which is compatible with it having snowed for 9 years.

It had not snowed in ten years

```
(s / snow
   :polarity -
   :range (t / temporal-quantity
             :quant 10
             :unit (y / year)))
```

4 Causal/Conditional/Concessive

4.1 Cause

The `cause` role is typically used for causal adverbial clauses such as *because* clauses. The `cause` role is used to annotate an answer to the question "why did the event happen?".

The wind broke the vase because it was fragile

```
(b / break
   :actor (w / wind)
   :theme (v / vase)
   :cause (f / fragile)
             :theme v))
```

A `cause` is one of two ways of representing the notion of a "reason" in WISeR, (See also `purpose` §4.2). In the following sentence, there are two reasons the judge sentenced the man.

The judge sentenced the man for speeding because he looked shifty

```
(s / sentence
   :actor (j / judge)
   :theme (m / man)
   :cause (a / and
             :op1 (s2 / speed
                      :actor m)
             :op2 (s3 / seem
                      :theme (s4 / shifty
                                 :theme m))))
```

WISeR also uses the inverse `cause-of` relation to represent some result states.[4]

The vase broke into pieces

```
(b / break
   :theme (v / vase)
   :cause-of (i / in-pieces
                :theme v))
```

[4]Notice we use `cause-of` instead of `end` here, since *pieces* is not something which is made out of the vase (i.e. a product). Moreover, it is not a grammatical argument of *break*. Finally, it does not take part in the material/product alternation which is indicative of the `end` relation.
i. He folded the paper into a card / He folded a card out of the paper
ii. He broke the vase into pieces / *He broke pieces out of the vase

He painted the house green

```
(p / paint
   :actor (h / he)
   :theme (h2 / house)
   :cause-of (g / green
                :theme h))
```

The soldiers marched themselves tired

```
(m / march
   :actor (s / soldier)
   :cause-of (t / tired
                :theme s))
```

4.2 Purpose

The role `purpose` is used to annotate an answer to the question "why was the event done?". A `purpose` is one of two ways to represent a "reason" in WISeR, (See also `cause` §4.1). In contrast to a `cause`, a `purpose` always follows the event.

She works for a living

```
(w / work
   :actor (s / she)
   :purpose (l / living))
```

She works to improve her life

```
(w / work
   :actor (s / she)
   :purpose (i / improve
               :theme (l / life
                         :poss h)))
```

A physical object may also have a `purpose`.

She found a trap for catching monkeys

```
(f / find
   :actor (s / she)
   :theme (t / trap
             :purpose (c / catch
                         :theme (m / monkey))))
```

4.3 Condition

The `condition` role is used for introducing an *if*-clause.

We will stay home if it rains

```
(s / stay
   :theme (w / we)
   :location (h / home)
   :condition (r / rain))
```

In combination with `polarity`, it can be used to represent an *unless* clause.[5]

We will win the tournament unless we lose the final game

```
(w / win
   :actor (w2 / we)
   :theme (t / tournament)
   :condition (l / lose
                 :polarity -
                 :actor (w2 / we)
                 :theme (g / game
                           :mod (f / final))))
```

It can also represent an unconditional *whether or not* clause.

We will go to the park whether it rains or not

```
(g / go
   :actor (w / we)
   :end (p / park)
   :condition (o / or
                 :op1 (r / rain)
                 :op2 (r2 / rain
                          :polarity -)))
```

These clauses are often fronted. In which case, use the inverse `condition-of` (See also `cause` §4.1).

If it rains, we will stay home

```
(r / rain
   :condition-of (s / stay
                    :theme (w / we)
                    :location (h / home)))
```

4.4 Concession

WISeR uses the role `concession` in the same way as AMR. It is used to represent concessive connectives such as *although* and *despite*.

[5]The AMR guidelines incorrectly places the negative polarity directly under the root concept, rather than embedded within the `condition`. Our example shows that this is incorrect. Consider the following sentences.

i. We will win the tournament unless we lose the final game.
ii. We won't win the tournament if we lose the final game.
iii. We will win the tournament if we don't lose the final game.

Both (ii) and (iii) would be true if (i) is true. However, (ii) would be true even if we cannot win the tournament with a draw. But (i) and (iii) would be false. This shows that (i) is closer in meaning to (iii) than (ii).

The game continued although it rained
The game continued despite the rain

```
(c / continue
   :theme (g / game)
   :concession (r / rain))
```

These clauses are often fronted, in which case you can use the inverse `concession-of`.

Although it rained, the game continued
Despite the rain, the game continued

```
(r / rain
   :concession-of (c / continue
                     :theme (g / game)))
```

Sometimes *but* is used concessively (see also `comparison` §5.6 for contrastive uses of *but*).

Trade has developed rapidly but it still has potential

```
(d / develop
   :theme (t / trade)
   :manner (r / rapid)
   :concession-of (h / have
                       :actor t
                       :theme (p / potential)
                       :mod (s / still)))
```

5 Mereology and Degrees

5.1 Domain and Mod

The roles `domain` and `mod` are inverses. The former is typically used in noun-copula-noun constructions.

They are birds

```
(b / birds
   :domain (t / they))
```

As well as in small clauses.

I consider him a friend

```
(c / consider
   :actor (i / i)
   :theme (f / friend
             :domain (h / he))
```

They are considered traitors

```
(c / consider
   :theme (p / person
             :domain (t / they)
             :actor-of (b / betray)))
```

The role `mod` is typically used for nominal modifiers such as adjectives.

Vice president

```
(p / president
   :mod (v / vice))
```

As well as relative clauses in which the main predicate is a noun (i.e., when you need to use the inverse of `domain`).

The man who is a lawyer

```
(m / man
   :mod (l / lawyer))
```

It is important to note, however, that `mod` is not used for all adjectives. Since the concept `toy` could be a `theme` of the concept `new` (not `domain`) we use the inverse of `theme`, `theme-of`, not `mod`.

The new toy

```
(t / toy
   :theme-of (n / new))
```

Likewise for `weather` and `cold`.

The cold weather

```
(w / weather
   :theme-of (c / cold))
```

Consider also the following more complicated example.

My favorite dog
The dog I favor

```
(d / dog
   :theme-of (f / favor
                :actor (i / i)))
```

5.2 Attribute

WISeR introduces the `attribute` role to account for a number of verb specific arguments, as well as providing a more intuitive description for some existing roles. The role `attribute` is used to annotate an argument which answers the question "In what respect does an argument have, or change in, the property described?".

Oftentimes, the `attribute` can appear redundant.

The man is short in stature

```
(s / short
   :theme (m / man)
   :attribute (s2 / stature))
```

The popcorn was free of charge

```
(f / free
   :theme (p / popcorn)
   :attribute (c / charge))
```

But this is not always the case.[6]

The man grew in courage

```
(g / grow
   :theme (m / man)
   :attribute (c / courage))
```

The man is rich in spirit

```
(r / rich
   :theme (m / man)
   :attribute (s / spirit))
```

Silver's advance in price

```
(a / advance
   :theme (s / silver)
   :attribute (p / price))
```

Attributes are commonly introduced by the prepositions *as* and *in*, and they add more specific information about some feature of one of the arguments (typically the `theme`). This includes non-result state secondary predicates.

The woman was accredited as an expert

```
(a / accredited
   :theme (w / woman)
   :attribute (e / expert))
```

The girl was denounced as a fraud

```
(d / denounce
   :theme (g / girl)
   :attribute (f / fraud))
```

[6]For the sentence *the man is rich in spirit*, PropBank would give *the man* the thematic role goal, and *spirit* the thematic role theme.

The girl employed the boy as a cleaner

```
(e / employ
   :actor (g / girl)
   :theme (b / boy)
   :attribute (p / person
                 :actor-of (c / clean)))
```

Lying counts as a sin

```
(c / count
   :theme (l / lie)
   :attribute (s / sin))
```

5.3 Quantity

The relation `quantity` is used to annotate numerical amounts.

Three boys passed the exam

```
(p / pass
   :actor (b / boy
             :quant 3)
   :theme (e / exam))
```

Several hundred apples

```
(a / apples
   :quant (s / several
             :op1 100))
```

Four out of five investors lost money

```
(l / lost
   :actor (p / person
             :actor-of (i / invest)
             :quant 4
             :subset-of (p2 / person
                           :actor-of (i2 / invest)
                           :quant 5))
   :theme (m / money))
```

It is also used to specify distance quantities and temporal quantities (See `extent` §5.5, `duration` §3.2 and `range` §3.4).

5.4 Degree

The `degree` role is used to introduce intensifiers like *very*, and *extremely* as well as "downtoners" like *somewhat* and *relatively*.

The girl is very tall

```
(t / tall
   :theme (g / girl)
   :degree (v / very))
```

The girl is too tall

```
(t / tall
   :theme (g / girl)
   :degree (t / too))
```

It is also used in comparatives and superlatives (See also `comparison` §5.6).

The girl is the best

```
(g / good
   :theme (g2 / girl)
   :degree (m / most))
```

5.5 Extent

The role `extent` is not to be confused with `degree`. This role is often used to quantify a predicate.

The road goes on forever

```
(g / go-on
   :theme (r / road
   :extent (f / forever))
```

The boy grew 3 inches

```
(g / grow
   :theme (b / boy
   :extent (d / distance-quantity
              :unit (i / inches
                       :quant 3)))
```

We will also use this relation to introduce a measure phrase in comparative constructions (See also `comparison` §5.6).

5.6 Comparison

The annotations in the AMR 3.0 corpus follow the suggestions in Bonial et al. (2018). As such, we adopt these suggestions for now, modulo the discarding of numbered `ARG`s.[7] Comparatives are represented using a reification of the `degree` relation, `have-degree`. Since WISeR does not use numbered `ARG`s, we introduce a `comparison` relation. The `comparison` role is given to arguments which something is being compared to or contrasted with.

[7] WISeR aims to seek potential improvements on this work in the future.

The girl is taller than the boy
The girl is taller than the boy is

```
(h / have-degree
   :theme (g / girl)
   :attribute (t / tall)
   :degree (m / more)
   :comparison (b / boy))
```

A full list of the relations used are as follows.[8]

`theme`	entity characterized by attribute
`attribute`	attribute (e.g. tall)
`degree`	degree itself (e.g. more/most, less/least, equal)
`comparison`	compared-to
`comparison`	reference to superset
`comparison`	consequence, result of degree

Table 2: List of arguments for `have-degree`

Below is an example of a superlative with a `comparison` argument.

The girl is the tallest of her friends

```
(h / have-degree
   :theme (g / girl)
   :attribute (t / tall)
   :degree (m / most)
   :comparison (f / friend
                  :poss g))
```

The following is an example of a 'degree consequence' construction.

The girl is too tall to sit comfortably

```
(h / have-degree
   :theme (g / girl)
   :attribute (t / tall)
   :degree (t2 / too)
   :comparison (s / sit
                  :theme g
                  :manner (c / comfort)))
```

Notice that the above sentence would typically be said when the girl is unable to sit comfortably (i.e., the consequence clause is non-veridical). However, rather than inserting negation or a modal concept here, Bonial et al. (2018) leave this representation as it is.[9] In later versions, WISeR aims to make improvements in this respect.

[8]Notice that we collapse three of Bonial et al.'s numbered `ARG` roles into one `comparison` role. We do this for several reasons: (i) they are all responsible for introducing a point of comparison, (ii) they never co-occur, (iii) the choice of numbered `ARG` depends entirely on the value of the `degree` role. As such, we may be able to get away with assuming that interpretation of `comparison` simply depends on the value of the `degree` (e.g., `more`, `most`, `too`, etc.).

[9]They note that sentences such as *the man was too drunk to drive* do not always entail that the man didn't drive.

Finally, we use `comparison` for certain verbal arguments, such as the second prototypical patient/theme argument assigned by PropBank to the verb *correlate*.

Life expectancy correlates with wages

```
(c / correlate
   :theme (e / expect
             :theme (l / live))
   :comparison (w / wage))
```

Likewise, PropBank assigns the first argument of *similar* an agent role and the second a patient/theme role. However, neither argument can reasonably be called an agent. The addition of `comparison` allows us to rectify this.

The girl is similar to the boy in height

```
(s / similar
   :theme (g / girl)
   :comparison (b / boy)
   :attribute (h / height))
```

We also use comparison for arguments of `contrast` and contrastive connectives such as *but* (following the annotation of contrastive *but* in AMR 3.0).

The boy likes it, but the girl does not.

```
(l / like
   :actor (b / boy)
   :theme (i / it)
   :theme-of (c / contrast
                :comparison (l2 / like
                                :actor (g / girl)
                                :theme i
                                :polarity -)))
```

5.7 Possession

The `poss` relation is used to represent ownership or possession.

He loved his children

```
(l / love
   :actor (h / he)
   :benefactive (c / children
                   :poss h))
```

Note that `poss` is different from `part-of`, as it shows ownership not the relationship between two parts of one thing.

The sailor's boat

```
(b / boat
   :poss (s / sailor))
```

The boat's sail

```
(s / sail
   :part-of (b / boat))
```

5.8 Part-of and Consist-of

We use `consist-of` to represent the substance which an instance of a concept is composed from.

The gold watch

```
(w / watch
   :consist-of (g / gold))
```

We can also use it to cover some verb specific roles such as that of `compose`.[10]

The team is composed of players

```
(c / compose
   :theme (t / team)
   :consist-of (p / player))
```

This can be read 'the composition of the team consists of players'.

The `part-of` relation is used when representing a part of an entity.

The engine of the car
The car's engine

```
(e / engine
   :part-of (c / car))
```

The boy's leg

```
(l / leg
   :part-of (b / boy))
```

The south of France

```
(s / south
   :part-of (c / country :wiki "France"
               :name (n / name :op1 "France")))
```

5.9 Subevent

The `subevent` relation is used to describe the larger event of which the event in question is a part. It is often introduced with the phrase *in which*.

[10]This receives the vn-role 'material' in the PropBank frame.

A massive bombardment in which 300 missiles rained on the capital

```
(b / bombard
   :mod (m / massive)
   :subevent (r / rain
                :theme (m / missiles
                          :quant 300)
                :location (c / capital)
                :direction (d / down))))
```

It contextualizes the event as part of an overarching event.

The speakers left on the final day of the conference

```
(l / leave
   :actor (s / speakers)
   :time (d / day
            :mod (f / final)
            :subevent (c / conference)))
```

6 Operators

The WISeR roles described in this section are adopted wholesale from the AMR guidelines. Annotators with experience converting text into AMR can safely skip this section.

6.1 Op

As in AMR, `opx` roles are used in conjunctions and disjunctions.

The boy and the girl swam

```
(s / swim
   :actor (a / and
             :op1 (b / boy)
             :op2 (g / girl))
```

As well as in spatial and temporal arguments.

The boy sang 10 minutes ago

```
(s / sing
   :actor (b / boy)
   :time (b2 / before
             :op1 (n / now)
             :quant (t / temporal-quantity
                       :unit (m / minutes)
                       :quant 10)))
```

And for named entities.

The Titanic

```
(s / ship
   :wiki "RMS_Titanic"
   :name (n / name)
            :op1 "Titanic"))
```

For more uses of `opx`, refer to the AMR guidelines.

6.2 Polarity

The `polarity` relation is used to evaluate the logical truth value of the statement and can be used to negate sentences. This relation is a binary value.

The boy doesn't go

```
(g / go
   :actor (b / boy)
   :polarity -)
```

This role negates the predicate under which it is immediately nested. Consider the following example in contrast to the first.

It is not the boy who goes

```
(g / go
   :actor (b / boy)
             :polarity -)
```

6.3 Polite

The `polite` role is used to annotate politeness markers. This role has a binary value.

We'd ask you to please leave

```
(a / ask
   :actor (w / we)
   :benefactive (y / you)
   :theme (l / leave
             :actor y
             :polite +))
```

6.4 Mode

The `mode` role describes the mood of the sentence and the intentions of the speakers. It can mark an imperative.

Let's go!

```
(g / go
   :actor (w / we)
   :mode imperative)
```

Wait here!

```
(w / wait
   :actor (y / you)
   :location (h / here)
   :mode imperative)
```

Or an expressive.

Wow!

```
(w / wow
   :mode expressive)
```

6.5 Example

The `example` role introduces something which is an example of a concept

The family vacations in resort spots like the beach

```
(v / vacation
   :actor (f / family)
   :location (s / spots
               :mod (r / resort)
               :example (b / beach)))
```

I like music such as country and rock

```
(l / like
   :actor (i / i)
   :theme (m / music
             :example (a / and
                         :op1 (c / country)
                         :op1 (r / rock))))
```

6.6 Name

The `name` role provides a concept's name.

The family's dog Snoopy barked

```
(b / bark
   :actor (d / dog
             :poss (f / family)
             :name (n / name
                      :op1 "Snoopy")))
```

6.7 Age

The `age` role provides an entity's age.

The 38 year old man injured his leg

```
(i / injure
   :actor (m / man
             :age (t / temporal-quantity
                     :quant 38
                     :unit (y / year)))
   :theme (l / leg
             :part-of m))
```

6.8 Value and Ord

The role `value` is used for specifying the numerical value of an entity.

Ninety-nine percent
99%

```
(p / percentage-entity
   :value 99)
```

While `ord` is used for ordinal numbers (i.e., 1st, 2nd, 3rd, etc.)

The second planet

```
(p / planet
   :ord (o / ordinal-entity
           :value 2))
```

6.9 Unit

The `unit` relation is used, often with `quantity`, to denote the measurement of a quantity.

She had planned her wedding for ten years

```
(p / plan
   :actor (s / she)
   :theme (w / wedding)
   :duration (t / temporal-quantity
                :quantity 10
                :unit (y / year)))
```

Units also don't have to be scientifically measured units.

a dozen bottles of water

```
(w / water
   :quantity (d / dozen)
   :unit (b / bottle))
```

We also must be explicit about what we are measuring when we use units. In the below example, without the `weight-quantity` predicate the meaning representation would be under specified.

The couple bought 4 pounds of rice

```
(b / buy
   :actor (c / couple)
   :theme (r / rice
             :quantity (w / weight-quantity
                          :quantity 4
                          :unit (p / pound)))
```

Similarly, we use `x-quantity` for other measurements such as volume for mass nouns.

The couple bought 2 gallons of milk

```
(b / buy
   :actor (c / couple)
   :theme (m / milk
             :quantity (v / volume-quantity
                          :quantity 2
                          :unit (g / gallon)))
```

6.10 List

The `list` relation is used to enumerate a list of items.

She believed she lived in the best city- one, everyone was friendly; two, the weather was perfect; and three, the food was delicious

```
(m / multi-sentence
   :snt1 (b / believe
            :actor (s / she)
            :theme (r / reside
                      :theme s
                      :location (c / city
                                   :mod (b / best))))
   :snt2 (f / friendly
            :list 1
            :actor-of (e / everyone))
   :snt3 (p / perfect
            :list 2
            :theme-of (w / weather))
   :snt4 (d / delicious
            :list 3
            :theme-of (f / food)))
```

7 Questions

WISeR uses the question tag `WISeR-question` to denote questions. For yes/no questions, `WISeR-question` is used in conjunction with the `polarity` relation to show that the truth value is in question.

Did the boy eat lunch?

```
(e / eat
   :actor (b / boy)
   :theme (l / lunch)
   :polarity (w / WISeR-question))
```

Does the teacher read a lot?

```
(r / read
   :actor (p / person
             :actor-of (t / teach))
   :frequency (f / frequent)
   :polarity (w / WISeR-question))
```

For *wh*-questions such as those containing *who*, *what*, *when*, *where*, *why*, and *how*, `WISeR-question` is used in the *wh*-item's argument position (e.g., *the boy ate what?*).

What did the boy eat?

```
(e / eat
   :actor (b / boy)
   :theme (w / WISeR-question))
```

How fast did the athlete run?

```
(r / run
   :actor (a / athlete)
   :manner (f / fast
              :degree (w / WISeR-question)))
```

Whose toy did the girl find?

```
(f / find
   :actor (g / girl)
   :theme (t / toy
             :poss (w / WISeR-question)))
```

Why did the baby cry?

```
(c / cry
   :actor (b / baby)
   :cause (w / WISeR-question))
```

For choice questions, we use `WISeR-choice` to denote options.

Do you want tea or coffee?

```
(w / want
   :actor (y / you)
   :theme (w2 / WISeR-choice
              :op1 (t / tea)
              :op2 (c / coffee)))
```

Did the teacher walk or did she drive to school?

```
(s / school
   :end-of (w / WISeR-choice
              :op1 (w2 / walk
                       :actor (g / girl))
              :op2 (d / drive
                      :actor t))))
```

Did the man win or lose the lottery?

```
(m / man
   :actor-of (w / WISeR-choice
                :op1 (w2 / win
                         :theme (l / lottery))
                :op2 (l2 / lose
                         :theme l)))
```

8 Relative Clauses

Relative clauses are represented with inverse roles.

The boy who wore red sang at the concert

```
(s / sing
   :actor (b / boy
             :actor-of (w / wear
                          :theme (r / red)))
   :location (c / concert))
```

The main predicate in this sentence is *sing* which therefore forms the root of our annotation. The predicate *wear red* is then introduced with the inverse relation `actor-of`.

The man saw the executive that moved into the large office

```
(s / see
   :actor (m / man)
   :theme (e / executive
             :actor-of (m / move
                          :end (o / office
                                  :mod (l / large))))
```

Note that the information about the executive moving into a large office is used to identify the person that the man hates. In this sentence, the man saw the executive. In contrast, the following sentence does not involve a relative clause.

The man saw that the executive that moved into the large office

```
(s / see
   :actor (m / man)
   :theme (m / move
             :actor (e / executive)
             :end (o / office
                    :mod (l / large))))
```

For this sentence to be true, the man need not directly see the executive. It is sufficient that he sees evidence that the executive is the new occupant of the large office.

9 Have-rel-role and have-org-role

WISeR follows AMR in using special predicate to attribute certain roles to people. For instance, a person who stand in a certain professional or personal relation to another.

she is my doctor

```
(h / have-rel-role
   :actor (s / she)
   :theme (i / i)
   :attribute (d / doctor))
```

`actor`	person who has role
`theme`	with whom
`attribute`	the relation

Table 3: List of arguments for `have-rel-role`

My girlfriend swims

```
(s / swim
   :actor (p / person
             :actor-of (h / have-rel-role
                          :theme (i / i)
                          :attribute (g / girlfriend))))
```

Other examples of `have-rel-role` include: *father, sister, husband, grandson, godfather, stepdaughter, brother-in-law, friend, boyfriend, buddy, enemy, landlord, tenant* etc.

We use a similar structure for `have-org-role`.

`actor`	person who has role
`theme`	organization
`attribute`	the role

Table 4: List of arguments for `have-org-role`

She is the company president

```
(h / have-org-role
   :actor (s / she)
   :theme (c / company)
   :attribute (p / president))
```

References

Bonial, C., Badarau, B., Griffitt, K., Hermjakob, U., Knight, K., O'Gorman, T., Palmer, M., and Schneider, N. (2018). Abstract meaning representation of constructions: The more we include, the better the representation. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Index