Annotation guidelines for the Block Game project

Barend Beekhuizen

v. 18 april 2012

1 Background

The aim of the Block Game project is to show that the extralinguistic context of a communicative situation can provide a child with a rich environment for learning linguistic semantics. In order to encode relevant features of the environment, I developed a set of annotations as described in these guidelines. Although the variables were selected with the game of the blocks and holes in mind, I believe them to be relatively general and applicable to other frames/events/scenarios the child finds herself in (eating, taking a bath). The variables are relatively low-level motoric and static relational predicates and simple object labels, none of which are expected to be beyond the cognitive development of a 16 month-old child.

2 Data

The data consists of videos of mothers and daughters (16 month-olds) playing a game of fitting the blocks in the holes of a bucket. There is considerable linguistic material relating to aspects of this game. The behavior consists of children trying to fit the blocks in and generally doing things with the toy (taking lid off, picking up blocks) mothers helping them and showing how it should be done. Every now and then, other actions take place, especially when the children get distracted.

3 Variables coded

The variables come in three regular tiers (**Behavior_mother**, **Behavior_daughter** and **Space**) and a fourth tier (**Dummy_variables**) where dummy objects can be coded (to be explained below). All tiers are coded within three-second windows of annotations, that is: we code on one string all actions taking place between, say, 0:12.000 and 0:14.999. Whenever an action takes place at the boundary of two coding windows, code it in that window in which the majority of temporal extension of the action takes place.

	0:00 - 0:03	0:03 - 0:06	...	n-3 - n
Behavior_mother	annotations$^1_1$	annotations $^1_2$	...	annotations$^1_{n/3}$
Behavior_daughter	annotations$^2_1$	annotations $^2_2$	...	annotations$^2_{n/3}$
Space	annotations$^3_1$	annotations $^3_2$	...	annotations$^3_{n/3}$
Dummy_variables	annotations$^4_1$	annotations $^4_2$	...	annotations$^4_{n/3}$

Table 1: Example of tiers and windows

3.1 Behavioral tiers

For the two behavioral tiers of annotation (**Behavior_mother** and **Behavior_daughter**), the annotations on the two behavioral tiers consist of a list of predicate-argument structures. The predicates come from a fixed set, the arguments can in principle be anything.

The syntax for writing behavioral predicate-argument structures is
$PREDICATE(ARGUMENT_1, ARGUMENT_2, \ldots, ARGUMENT_n)$.
That is, the arguments follow the predicate, enclosed by parentheses and separated with a comma. Multiple predicate-argument structures are separated from each other with a space. Do not use tabs, carriage returns or other white space markers for spaces, nor colons, semicolons or anything else for commas, nor other brackets than round ones.

3.1.1 Predicates

The predicates coded are given in table 8.

The predicates each have thematic roles that denote who does what to whom. The **agent** is the volitional starting point of some action; he initializes it and perpetuates it. As the agent is always either the mother or the daughter (depending on the tier), we can save some keystrokes by leaving this argument out in the annotations. The **theme** is the participant that undergoes the agent's actions. Several locational roles exist as well (**source,goal,direction**). These denote locations of the theme at different points of the action. The instrument role is optional and denotes the situation in which the agent is not directly manipulating the object with its hands, but rather has some intermediate object to do so. Holding food on a fork, hitting a bal with a club and pushing one block with another block would be examples of relations involving instruments.

In the first 8 predicates, we use the notion of **manual control** to denote the situation in which the agent can do whatever it wants with the theme (it has 'control' over the theme) and the theme is located in the hand of the agent or adjoined to an object functioning as an instrument, that is in the hand of the agent. In predicates 1, 9 and 10, there is no manual control.

Some predicates imply an other predicate. If such a predicate is coded in a certain window, the implied predicate, when applying to the same arguments does not have to be coded. Conversely, the implied predicate is only coded if none of the predicates implying that predicate hold. The following implication relations hold:

grab	implies	**reach**
move	implies	**hold**
position	implies	**hold**
hit	implies	**hold**
show	implies	**hold**
letgo	implies	**hold**
force	implies	**letgo**

Table 2: Implication relations between predicates

Some predicates look very similar from a behavioral point of view. In both pointing and reaching, an arm is extended towards an object. In both showing and moving+holding, an object is moved and then held at a point. The difference, obviously lies in the intentionality: pointing and showing are communicative acts, reaching, moving and holding are acts of object manipulation. However, intentions are unobservable states, and to warrant the principle that what we code is behavior, we can use the following decision procedure for each pair

3.1.2 Arguments

The arguments in the behavioral predicates come in two types, viz. Objects (subscript-O) and Spaces (subscript-S). Objects are categorical descriptions of an object at the most specific level of description. Spaces contain a type of relation coupled with an object. Note that type of argument and role are to some extent orthogonal: the theme role of a reach predicate can be a location (Space) or an Object, and the goal role of a force predicate can be both too (towards an object in general, or towards a spatial relation with that object).

question	*if so*	*else*
Is the behavior of this participant followed by a **grab** with the same theme?	**reach**	next question
Is the behavior of this participant directly followed by or simultaneous with the child looking at the face of the mother?	**point**	next question
Is the behavior of this participant directly followed by or simultaneous with the child looking at the theme?	**reach**	**unclear**

Table 3: Deciding between **reach** and **point**

question	*if so*	*else*
Is the goal of the motion or the current location of the theme *between* the fronts of the child and the mother?	next question	**move/hold**
Is the behavior of this participant directly followed by or simultaneous with the participant looking at the face of the other participant?	**show**	next question
Is the behavior of this participant directly followed by or simultaneous with the participant looking at the theme?	**reach**	next question
Is the behavior of this participant followed by the other participant's **grab** with the same theme?	**show**	**unclear**

Table 4: Deciding between **move/hold** and **show**

Spatial arguments Spatial arguments always have the theme of the argument structure as the **trajector**, that is: as the participant that is conceptualized as foregrounded with respect to a background. The background or **landmark** then, is described as the complement of the Space-argument. We only code the four relations in table 5. Space-arguments are coded as follows: $PREDICATE - LANDMARK$. The spatial relations take as a landmark any object as described in the following paragraph. Sometimes an object is simply in a participant's hand in the air. In those cases we code the location as **in-air**.

	code	*description*
1	**in**	containment of the trajector in the landmark
2	**on**	horizontally supported contact between the trajector and the landmark
3	**at**	other types of physical contact between the trajector and the landmark
4	**near**	adjacency without physical contact

Table 5: Spatial relations

Object arguments and complements of Spatial arguments Anything can in principle be an object argument of a behavioral predicate, but several aspects of the block game and its environment we encode recur often and therefore warrant a uniformized description. For all other objects (dogs, cups, other toys), simply use a consistent, uniquely identifying code consisting of lowercase letters only and no spaces. A toy rabbit may thus be "toyrabbit", the cup of the observer "observercup" and so on.

The toy consists of a bucket, with a lid and a handle, and contains twelve blocks. These blocks come in four colors (red, yellow, green, blue) and four shapes (circle, square, triangular and star-shaped). See the reference sheet for artist's renderings of the toy. The shapes of the blocks correspond to four holes in the lid. The game is played by a mother and a daughter and is observed by an observer. The game is furthermore played on a surface, like a table or a floor. The annotation codes for these elements are:

The code for the specific types of blocks is composed out of a prefix **b**, two letters for the color, and two letters for the shape, so **bresq** is a red square. The following are the color-shape codes:

	code	*description*
1	**to**	toy
1.1	**bu**	bucket
1.2	**ha**	handle
1.3	**li**	lid
1.3.1	**hoci**	circular hole
1.3.2	**hosq**	square hole
1.3.3	**hotr**	triangular hole
1.3.4	**host**	star-shaped hole
2	**mo**	mother
3	**ch**	child
4	**ob**	observer
5	**table**	table
6	**floor**	floor
7	**air**	air

Table 6: Fixed codes for frequent arguments

prefix	*code*	*color*	*code*	*shape*
b	**re**	red	**ci**	circular
	gr	green	**sq**	square
	bl	blue	**tr**	triangular
	ye	yellow	**st**	star-shaped

Table 7: Fixed morphology for blocks and their attributes.

Often, you will not immediately recognize the shape or color of a block or a hole. Then we can use dummy variables. See section 3.3 for instructions.

Sometimes, a behavioral predicate does not have a specific block as its goal (e.g. reaching, especially by children). In such a case, the generic variable **blocks** may be used.

Code as specific as possible. When a predicate applies to a part of an object rather than the whole, use the code for the part (e.g. **grab** applied to a **ha**ndle theme, instead of a **to**y theme).

3.2 Spatial tier

In the spatial tier, we code relative spatial relations between several objects related to the game. The spatial relations we code are simply whether a block is inside (**in**) of or outside of (**out**) the bucket and whether the lid is on (**on**) or off (**off**) the bucket. The syntax for these predicates is $TRAJECTOR - PREDICATE$ (so: **li-on** means that the lid is on the bucket, and **byesq-out** means that the yellow square block is now outside of the bucket. For the blocks, the labels as mentioned before (color+shape) are used.

When all blocks change their position w.r.t. the bucket, we do not describe all of them separately, but use the generic term **blocks** instead (e.g. in the initial state, we always find **blocks-in**).

3.3 Dummy variable tier

Sometimes, an object is not immediately recognizable (especially the blocks), and so the coder will have to resort to either browsing through the video to resolve the identity of the block, or leaving the action uncoded at all. To avoid this, you can use the prefix code for that object (**ho** for holes, **b** for blocks and **o** for other objects), followed by an unique number $1..n$. To keep track of these dummies, you code them on the fourth tier, viz. the **Dummy_variable** tier. Multiple dummy variables are separated with a whitespace.

	predicate	*arguments*	*description*
1	**reach**	(theme$_{O/S}$, [instrument$_O$])	an agent extends its hand or an instrument towards a theme with the intention of grabbing it, but does not yet touch it. See the decision table below.
2	**grab**	(theme$_O$, [instrument$_O$])	an agent brings a theme under its manual control by positioning its hand around (part of) the theme and/or squeezing, or by applying an instrument that brings the theme under the agent's manual control.
3	**show**	(theme$_O$, recipient$_O$, [instrument$_O$])	an agent positions a theme under the agent's manual control so as to draw the recipients attention to that theme. See the decision table below.
4	**position**	(theme$_O$, location$_S$, [instrument$_O$])	an agent moves a theme under the agent's manual control vis-a-vis a location where the distance between theme and location does not change significantly.
5	**move**	(theme$_O$, source $_S$, goal$_S$, [instrument$_O$])	an agent moves a theme under the agent's manual control from a source location to a goal location where the distances between the theme and the source and goal respectively differ significantly during the process. See the decision table below.
6	**hit**	(theme$_O$, [instrument$_O$])	the agent or an instrument under the agent's manual control forcefully touches a theme upon impact.
7	**hold**	(theme$_O$, [instrument$_O$])	an agent has manual control over a theme by squeezing the theme, having its hand(s) around it or being perpetually attachet to it by means of an instrument. See the decision table below.
8	**letgo**	(theme$_O$, [instrument$_O$])	an agent ceases to have manual control over a theme by no longer squeezing the theme, holding its hand around it or keeping perpetual attachment via an instrument.
9	**force**	(theme$_O$, source $_S$, goal $_{O/S}$ [instrument$_O$])	an agent exerts force on the theme so that the theme moves and afterwards no longer is under the agent's control if it was so before.
10	**point**	(theme$_O$, recipient$_O$, [instrument$_O$])	an agent extends its hand towards a theme so as to draw a recipients attention towards the theme. See the decision table below.
10	**other**	(descriptions)	a clear manual action takes place which cannot be categorized as predicates 1-10. Write down your description of the action as the argument of the other-predicate.
11	**unclear**	(predicates1..n)	one or more of predicates 1-10 apply, but the coder is not certain which one.
12	**oov**	()	predicate applies when the agent's hand are out of the camera's view for the majority of the window.

Table 8: Behavioral predicates

If you resolve a dummy variable later on in the procedure, you go back to the place on the Dummy_variable tier where you introduced it, and write "=<CODE>" directly following to it, where <CODE> is the actual object code.

3.4 Example

1.	**behavior_mother**	point(hosq,ch)
	behavior_child	move(byesq,near-hoci,near-host) position(byesq,host) hold(b2)
	space	
	dummy_variables	
2.	**behavior_mother**	
	behavior_child	move(b3,near-hoci,near-hosq) letgo(b3) hold(b2)
	space	b3-in
	dummy_variables	b3=bblst b2=bgrsq
3.	**behavior_mother**	
	behavior_child	letgo(bgrci) hold(b2) grab(b4)
	space	bgrci-in
	dummy_variables	b4=byest
4.	**behavior_mother**	move(li,on-bu,on-floor) show(bu,ch)
	behavior_child	move(b3,on-floor,near-hotr) grab(hotr) letgo(hotr) hold(b2)
	space	li-off
	dummy_variables	
5.	**behavior_mother**	move(ch,in-air,on-ground) force(door,cupboard)
	behavior_child	grab(li)
	space	blocks-in li-on
	dummy_variables	

4 Annotation procedure

For coding the fragments, we use the video annotation tool ELAN (Max Planck Institute for Psycholinguistics, Nijmegen, http://www.lat-mpi.eu/tools/elan/).

4.1 Initializing an annotation file

To start annotating, follow these steps:

- Start ELAN.
- Turn the volume all the way down to zero.
- Go to **File**, click **New** (Ctrl+N). A window for selecting the files pops up.
- Select the video file you want to work on and add it with the >> button.
- Toggle **Template** and select the template with your initials attaches (e.g. **template_blockgame_bb.etf**); add it with the >> button.
- Click **OK** to return to the main window
- Save the annotations as the name of the video, followed by the .eaf extension (e.g. **mi014.eaf**). (Ctrl+Shift+S)
- Go to **Tier**, click **Create Regular Annotations**. A window for creating regular annotations pops up.
- Resize the window if necessary, select all tiers, and set
  - **Start time** to 00:00:00.000

- **Duration** to 00:00:00.000
- **End time** to 00:10:00.000
- **Annotation size** to 00:00:03.000

Having followed these steps, you will see that all four tiers are now split into windows of three seconds.

4.2 Annotating

Now, we can annotate the tiers one at a time. Starting with **Behavior_mother**, and working from the beginning of the video to the end:

- View three seconds at a time by
  - clicking somewhere inside of the tier, thereby selecting that three-second fragment,
  - clicking the ▷**S** symbol or pressing **Shift+space**.
- Once the section has finished playing, the crosshair will return to the starting point of the section.
- By double clicking inside the three-second area of the tier, you can write annotation codes (cf. section 3).
- Importantly, the coded variables are only stored after pushing **Ctrl+Enter**, and disappear when only pushing Enter.
- The arrow buttons to the right of the ▷**S** symbol or **Alt+arrow** on your keyboard allow you to go to the next and previous 3-second fragment in the video.

After Behavior_mother, we code **Behavior_child**, and finally **Space** in the same way.

If you do not see these buttons, you might be in the wrong "mode". To check, select the Options menu, and make sure you are in the Annotation Mode. Don't forget to save often!

The reference sheets in this manual can be kept at hand as a reminder of the categories and a means of tie-breaking in case of doubt. Furthermore, I would like you to keep a log of encountered problems during the annotation (software related, concerning variables, etc.). Writing materials are provided in the top drawer of the desk and the notes should be digitized at the end of each coding session.

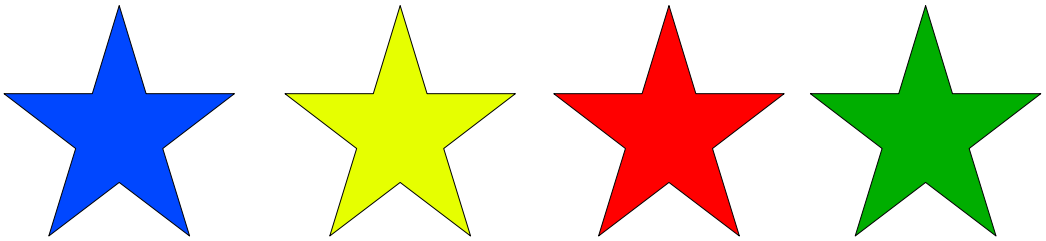
4.3 Rounding up an annotation

After you have annotated an entire video,

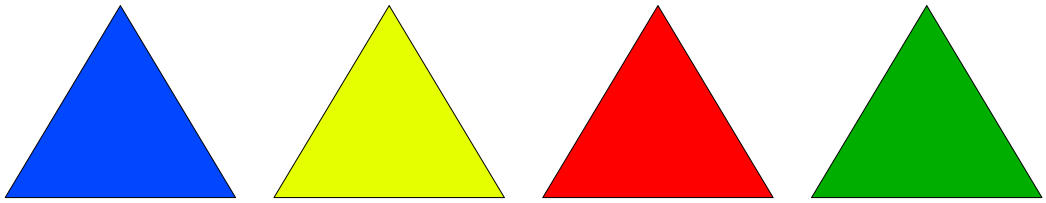
- you save the .eaf file
- you close ELAN
- you open a terminal screen (Applications > System > Terminal)
- you navigate to the directory /blockgame/scripts (type: `cd blockgame/scripts`).
- you run the script that will check the file you were working on for inconsistencies by typing
  `perl annotation_checker.plx` $< FILE >$`.eaf`
  where $< FILE >$.eaf should be the name of the file you annotated.
- you follow the instructions on the screen and when asked to reply, finish your reply by hitting enter.
- you close the terminal screen
- you open the log-file (`log_`$< NAME >$`.txt`) where $< NAME >$ stands for your initials and you fill in the details.

5 Reference sheets

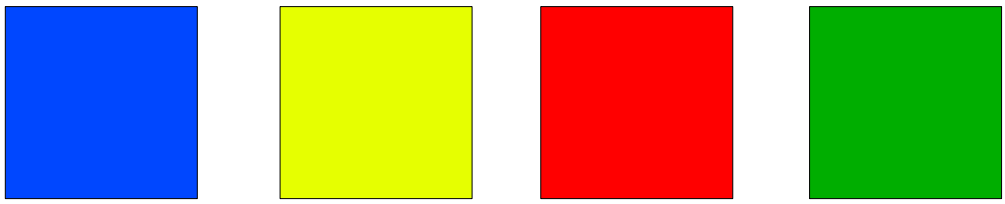
THE BLOCKS



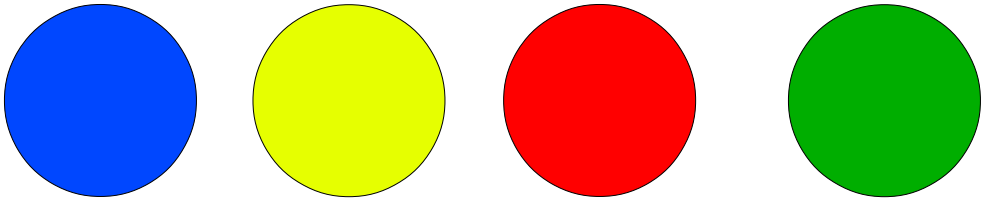
bblst byest brest bgrst



bbltr byetr bretr bgrtr

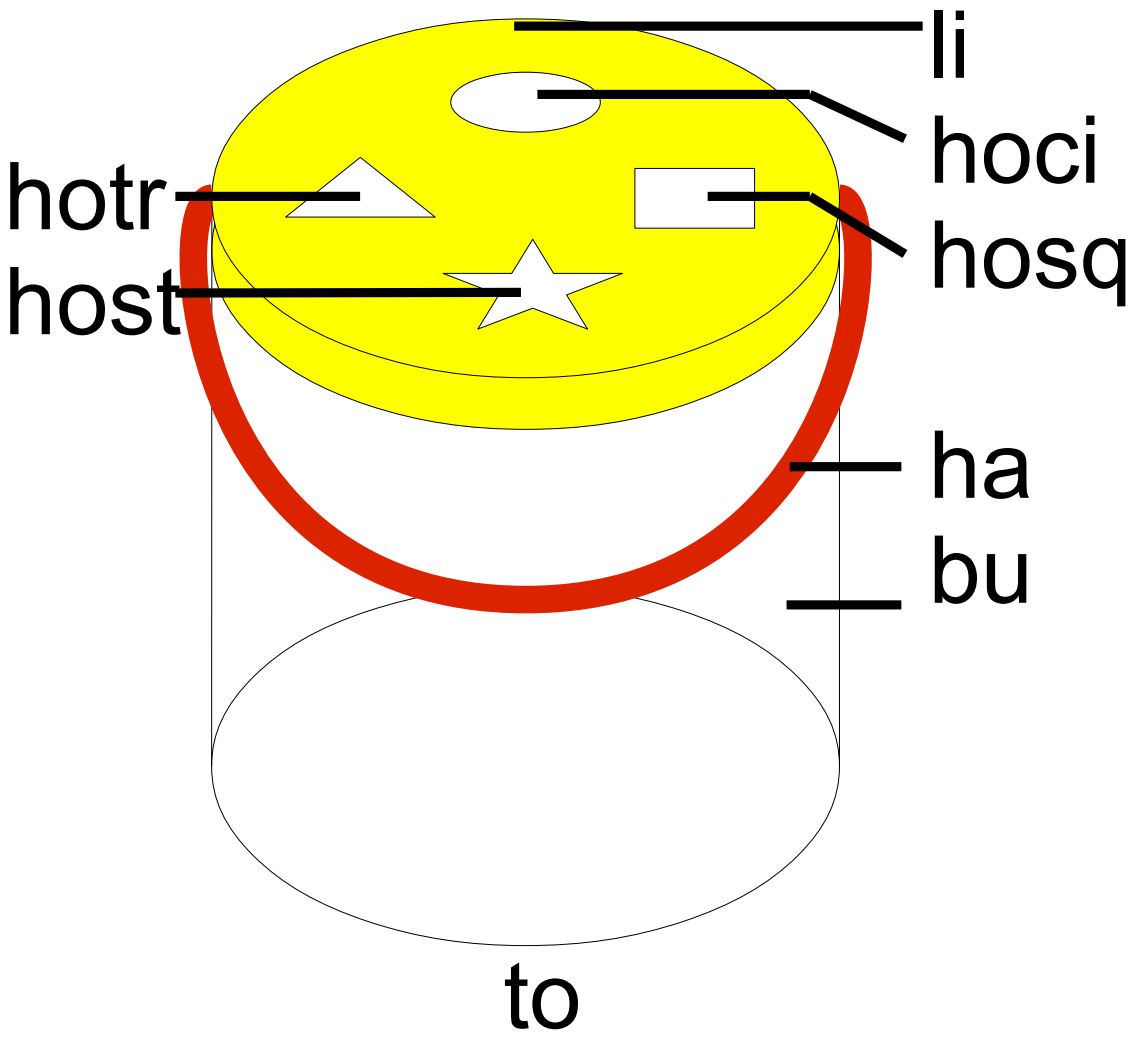


bblsq byesq bresq bgrsq



bblci byeci breci bgrci

THE TOY



OTHER ARGUMENTS

mo	mother
ch	child
ob	observer
floor	floor
table	table
air	air

PREDICATES

SPATIAL

bx-in / bx-out
for blocks

li-on / li-off
for the lid

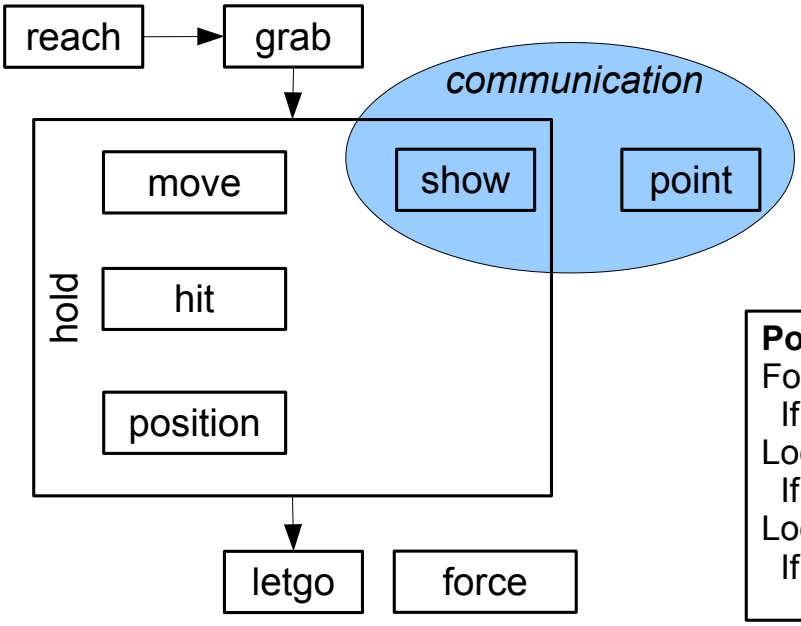
show vs **move/hold**
between ch&mo?
  if not --> **move/hold**
looking at other?
  if so --> **show**
Looking at theme?
  If so --> **move/hold**
Followed by grab?
  If so --> **show**

BEHAVIORAL

force (Th_o, Dir_s, [l_o])
grab (Th_o, [Ins_o])
hit (Th_o, [Ins_o])
hold (Th_o, [Ins_o])
letgo (Th_o, [Ins_o])
move(Th_o, So_s, Go_s, [Ins_o])
point (Th_o, Rec_o, [Ins_o])
position(Th_o, Loc_s, [Ins_o])
reach ($Th_{o/s}$, [Ins_o])
show(Th_o, Rec_o, [Ins_o])

other unclear oov

SPATIAL ARGUMENTS: { in-LM, on-LM, at-LM, near-LM }



Point vs **reach**
Followed by grab?
  If so --> **reach**
Looking at other?
  If so --> **point**
Looking at theme?
  If so --> **reach**