Percepcja i motoryczność w ACT-R

Konrad Miazga

Poznan University of Technology

Poznań, 2018

Queries

Oprócz kierowania zapytań do bufora (np. prośba o udostępnienie chunku) możemy również go odpytywać o jego stan. Takie specjalne zapytania o stan bufora będziemy nazywać *query*. *Query* będziemy stosować w lewej części (warunku) reguł produkcji.

Składnia *query* jest podobna do zapytań już nam znanych. Zwróć uwagę, iż tym razem nazwę bufora poprzedza symbol „?”:

Przykład

```
?retrieval>
        buffer full
```

Powyższe zapytanie zwróci TRUE, jeżeli w buforze retrieval znajduje się obecnie jakiś chunk lub FALSE gdy jest pusty.

Queries – stan bufora

Możemy zadawać również inne zapytania o stan bufora, zmieniając symbol w slocie *buffer* zapytania. Poniżej znajdują się inne wartości slotu *buffer* które powinny działać dla dowolnego bufora:

- **full** – TRUE jeżeli w buforze znajduje się obecnie jakiś chunk.
- **failure** – TRUE jeżeli dla bufora stwierdzono wystąpienie błędu.
- **empty** – TRUE jeżeli bufor jest pusty i nie stwierdzono wystąpienia błędu.
- **requested** – TRUE jeżeli *full* albo *failure* i są one efektem zapytania do bufora.
- **unrequested** – TRUE jeżeli *full* albo *failure* i **NIE** są one efektem zapytania do bufora.

Queries – stan modułu

Możemy również pytać się bufora o stan jego modułu. Tym razem będziemy musieli użyć jednak slotu *state*. Poniżej znajdują się inne wartości slotu *state* które powinny działać dla dowolnego bufora:

- **busy** – TRUE jeżeli moduł wykonuje obecnie jakieś działanie.
- **free** – TRUE jeżeli moduł **nie** wykonuje obecnie żadnego działania.
- **error** – TRUE jeżeli od czasu poprzedniego zapytania do tego bufora, w jego module wystąpił błąd.

Queries – specyficzne zapytania

Niektóre bufory będą pozwalały również na specyficzne dla siebie zapytania, np. dla bufora *retrieval* możemy użyć:

```
?retrieval>
        recently-retrieved t
```

Takie query zwróci TRUE jeżeli chunk znajdujący się obecnie w buforze był już w nim kiedyś wcześniej.

ODCZYTYWANIE LITERY Z EKRANU

Zadanie 1a

Załaduj do ACT-R eksperyment *demo2.lisp* i uruchom go za pomocą polecenia:
(do-demo2 'human)
W ten sposób uruchomisz eksperyment w lispie w trybie interakcji z użytkownikiem (w tym wypadku – tobą). Twoim zadaniem jest wciśnięcie na klawiaturze litery która została wyświetlona na ekranie.

Plik z modelem:
...\ACT-R Standalone Environment\tutorial\unit2\demo2.lisp

Zadanie 1b

Uruchom nasz model ponownie, tym razem w trybie interakcji z modelem: *(do-demo2)*
Zwróć uwagę na czerwone kółko – wskazuje ono miejsce na który skupia obecnie uwagę nasz model.

Spójrz na wyprodukowane przez eksperyment logi: jakie pojawiły się w nim nowe moduły i bufory?

Analiza logów (lisp)

Przykładowa linia logu

0.000 GOAL SET-BUFFER-CHUNK GOAL FIRST-GOAL REQUESTED NIL

- 0.000 – czas zdarzenia (symulowany),
- GOAL – moduł inicjujący zdarzenie,
- SET-BUFFER-CHUNK – zdarzenie, w tym wypadku akcja wstawienia chunka do bufora,
- GOAL – bufor z którym związane jest zdarzenie,
- FIRST-GOAL – nazwa użytego chunka,
- REQUESTED NIL – zdarzenie nie zostało zlecone przez żadną regułę produkcji.

Nowe moduły

W ramach dzisiejszych zajęć zapoznamy się z trzema nowymi modułami:

- Moduł wzrokowy (Vision)
- Moduł motoryczny (Motor)
- Moduł wyobrażeniowy (Imaginal)

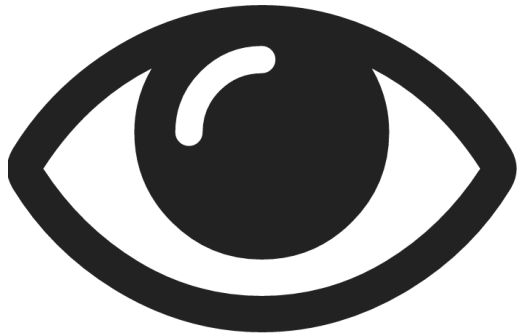
Moduły wzrokowy i motoryczny zostały opracowane przez Mike'a Bryne (obecnie Rice University).

Moduł wyobrażeniowy (Imaginal)

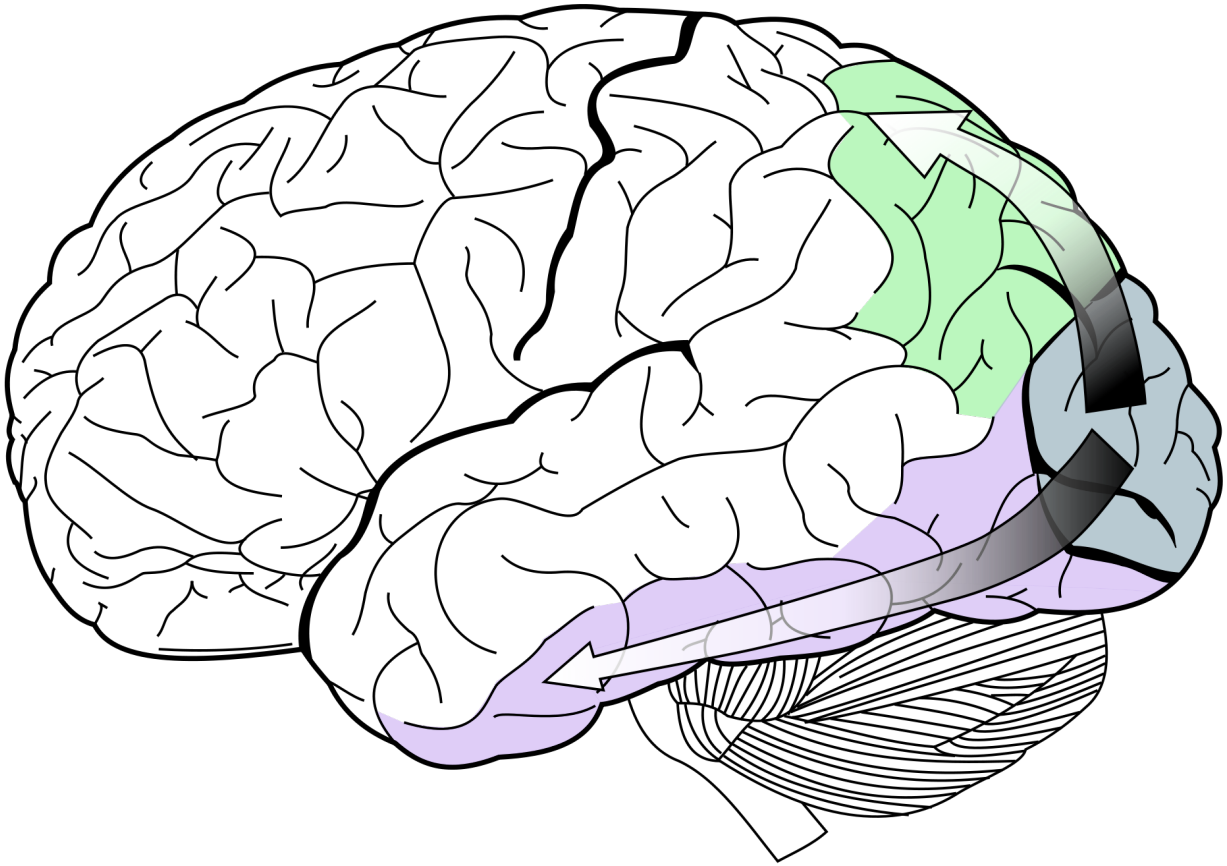
- Posiada bufor Imaginal
- Zapytania do niego **tworzą** nowe chunki (takie jak podane w zapytaniu) i wstawiają je do bufora Imaginal.
- Stworzenie nowego chunka (reprezentacji nowej wiedzy) zajmuje jednak czas – domyślnie ustawione 0.2s

Moduł wzrokowy (Vision)

- Posiada dwa bufory: Visual oraz Visual-Location.
- Służy do znajdowania oraz rozpoznawania treści znajdujących się na ekranie (w oknie eksperymentu).
- Same procedury znajdowania / rozpoznawania elementów nie są modelowane w ACT-R.
- W związku założeniem o równoległości działania modułu wzrokowego i reszty modelu, zapytania do bufora Visual-Location realizowane są natychmiastowo.
- Procesy związane z buforem Visual zajmują jednak trochę czasu (np. przesunięcie uwagi i rozpoznanie symbolu zajmują 0.85s).



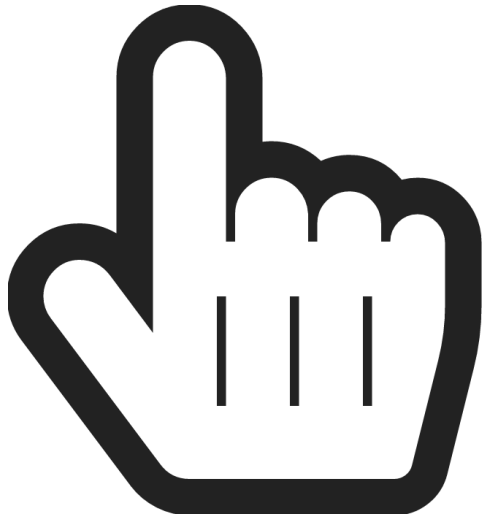
Moduł wzrokowy (Vision)



Rysunek: Zielony – ścieżka WHERE – bufor Visual-Location. Fioletowy – ścieżka WHAT – bufor Visual.

Moduł motoryczny (Motor)

- Posiada jeden bufor Manual.
- Moduł ten ogranicza się do sterowania ruchem rąk agenta, a więc pozwala na podejmowanie czynności takich jak wciskanie klawiszy klawiatury czy też sterowanie myszką.
- Bufor manual nie będzie nigdy zawierał żadnego chunku! Służy on jedynie do komunikacji się modułem motorycznym.
- Moduł ten posiada więcej stanów niż tylko *busy free* i *error*, ale nie będziemy o nich wspominać.



Przebieg działania modelu

Nasz model będzie obsługiwał eksperyment w prosty, liniowy sposób. Po kolei będą się odpalały cztery reguły produkcji:

1. Znajdź nową literę (*find-unattended-letter*)
2. Obsłuż literę (*attend-letter*)
3. Odkoduj literę (*encode-letter*)
4. Zareaguj (*respond*)

Na następnych slajdach omówimy działanie tych reguł oraz wszystko co się z nimi wiąże.

Znajdź nową literę (*find-unattended-letter*)

```
(P find-unattended-letter
=goal>
ISA read-letters
state start
==>
+visual-location>
:attended nil
=goal>
state find-location
)
```

W linijce `:attended nil` odwołujemy się do ***parametru*** zapytania. Parametry działają jak sloty, z tą różnicą, że nie są one zdefiniowane dla chunków i działać będą dla dowolnego zapytania do bufora. Każdy bufor może jednak przyjmować inne parametry.

Tutaj, prosimy bufor *Visual-Location* o dostarczenie nam takiej lokacji na którą jeszcze nie „patrzyliśmy” (**nil**). Moglibyśmy również poprosić o lokację już wcześniej odwiedzoną (**t**) lub taką, która dopiero niedawno się pokazała (**new**).

W ramach omówionego zapytania do bufora *Visual-Location* otrzymamy poniższy chunk opisujący naszą lokację:

VISUAL-LOCATION0-0-1

VALUE	TEXT
COLOR	BLACK
HEIGHT	10
WIDTH	7
SCREEN-X	130
SCREEN-Y	156
DISTANCE	1080
KIND	TEXT
SIZE	0.19999999

Screen-X oraz Screen-Y opisują odległość lokacji od lewego górnego narożnika okienka. Zazwyczaj nie będziemy się jednak odwoływać do lokacji za pomocą współrzędnych a raczej pewnych opisów położenia.

Obsłuż literę (*attend-letter*)

```
(P attended-letter
=goal>
ISA read-letters
state find-location
=visual-location>
?visual>
state free
==>
+visual>
cmd move-attention
screen-pos =visual-location
=goal>
state attend
)
```

W linijce `=visual-location>` nie definiujemy żadnych wartości slotów. Jedyna jej rola to upewnienie się, że w buforze *Visual-Location* znajduje się jakikolwiek chunk.

Jamming

W linijkach

```
?visual>
state free
```

sprawdzamy czy moduł wzrokowy nie przeprowadza obecnie żadnych operacji. Dlaczego?

Większość modułów może obsługiwać tylko jedno zapytanie na raz. Gdybyśmy skierowali nasze zapytanie z prawej strony produkcji do bufora gdy moduł jest zajęty, groziłby nam ***jamming***.

Zachowanie w razie *jammingu* różni się między modułami: mogą one zignorować nowe zapytanie, bądź przerwać wykonywanie poprzedniego. O ile możemy, należy zawsze unikać ryzyka wystąpienia *jammingu*!

```
+visual>
cmd move-attention
screen-pos =visual-location
```

Do bufora *Visual* możemy kierować różne rodzaje zapytań. Rodzaj zapytania zależy od podanej komendy (**cmd**). W linijce `+visual>` nakazujemy przesunięcie uwagi modułu do nowej lokacji. Inną możliwą komendą byłoby **clear** w wyniku którego, moduł *Vision* przestałby zwracać uwagę na ekran.

Nowa lokacja podana jest w slocie **screen-pos**. Linijka `screen-pos =visual-location` informuje bufor, iż wartość **screen-pos** należy pobrać z bufora *Visual-Location*.

W ramach omówionego zapytania do bufora *Visual* otrzymamy poniższy chunk opisujący symbol z naszej lokalizacji:

```
TEXT0-0
    SCREEN-POS VISUAL-LOCATION0-0-0
    VALUE "v"
    COLOR BLACK
    HEIGHT 10
    WIDTH 7
    TEXT T
```

Dzięki slotowi **text** o wartości **t**, wiemy iż w naszej lokalizacji znajduje się tekst. Jego treść możemy znaleźć w slocie **value**: odczytana litera to „v”.

Odkoduj literę (*encode-letter*)

```
(P encode-letter
=goal>
ISA read-letters
state attend
=visual>
value =letter
?imaginal>
state free
==>
=goal>
state respond
+imaginal>
ISA array
letter =letter
)
```

W linijce `?imaginal>...` upewniamy się, że nasze zapytanie do modułu *Imaginal* nie spowoduje *jammingu*.

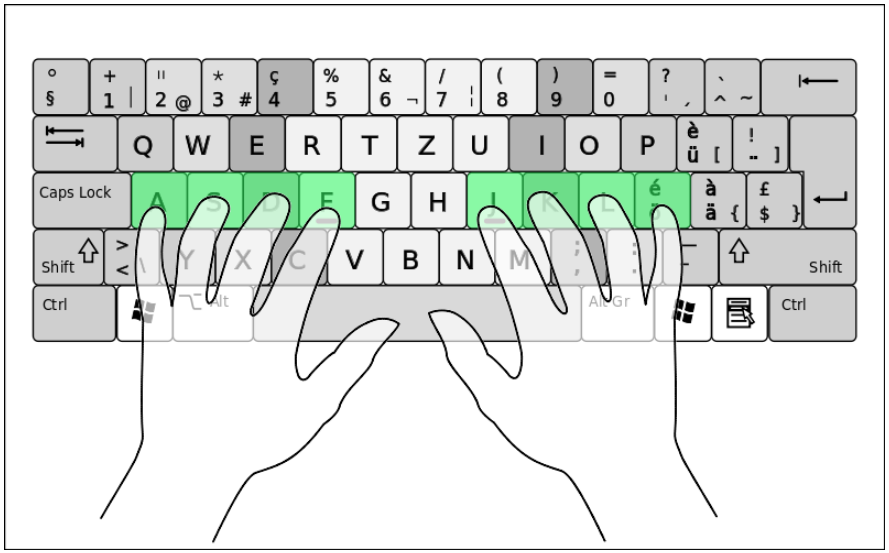
Zareaguj (*respond*)

```
(P respond
=goal>
ISA read-letters
state respond
=imaginal>
ISA array
letter =letter
?manual>
state free
==>
=goal>
state done
+manual>
cmd press-key
key =letter
)
```

W linijce `?manual>...` upewniamy się, że nasze zapytanie do modułu *Manual* nie spowoduje *jammingu*.

Podobnie jak do bufora *Visual*, do bufora *Manual* możemy również kierować różne zapytania. W tym wypadku, jest to komenda **press-key**. **press-key** modeluje akcję wciśnięcia pewnego klawisza **key** na klawiaturze. Modelowany jest tu sprawny (ale nie wybitny) użytkownik klawiatury:

- Zaczyna on z palcami na rzędzie domowym:



- Przesuwa on odpowiedni palec na odpowiedni klawisz (bez patrzenia).
- Wciska on klawisz.
- Cofa palec do pozycji pierwotnej.

Opóźnienia w module motorycznym

Każda ze wspomnianych czynności będzie zabierała trochę czasu.
Spójrzmy do logów:

Czas	Log	Akcja
0.485	MOTOR PRESS-KEY KEY v	Otrzymanie zlecenia
0.735	MOTOR PREPARATION-COMPLETE	Przygotowanie ruchu
0.785	MOTOR INITIATION-COMPLETE	Zainicjowanie ruchu
0.885	MOTOR OUTPUT-KEY #(4 5)	Wciśnięcie klawisza
1.035	MOTOR FINISH-MOVEMENT	Powrót palca

Zadanie 2

Przetestuj opóźnienia związane z przesuwaniem palców, modelowane przez moduł *Motor*. Sprawdź czasy dla następujących symboli:
B, C, F, 6, +, =

Czy zaobserwowałeś różnicę w czasach wykonań eksperymentu? Z czego one wynikają?

Aby móc kontrolować wyświetlany symbol, stwórz kopię pliku demo2. Następnie, zamień w niej linijki 18 — 22 na następujące:
(let* ((text1 "B")
W tym wypadku wyświetlona zostanie litera „B”.

Strict harvesting

Jeśli prześledzisz log wykonania eksperymentu *demo2*, zauważysz, że czasami następowało czyszczenie bufora z lewej strony produkcji, pomimo że po prawej stronie nie kierowaliśmy do niego żadnego zapytania. Wynika to reguły *strict harvesting* która domyślnie działa dla wszystkich buforów oprócz *Goal* i *Temporal*.

Jeżeli w pojedynczych wypadkach chcemy uniknąć czyszczenia bufora w wyniku testu, możemy po prawej stronie produkcji dokonać pustej modyfikacji, np.:

Pusta modyfikacja

```
=visual>
```

Visual re-encoding

W logach wykonania eksperymentu można ujrzeć jeszcze jedną niespodziewaną rzecz, mianowicie linijki:

```
0.970 VISION Encoding-complete VISUAL-LOCATION0-0-1 NIL
0.970 VISION No visual-object found
```

Pod koniec wykonania programu, moduł *Vision* bez pytania dokonuje próby zinterpretowania symbolu w naszej lokacji! Czemu?

Wiąże się to z pojęciem *visual re-encoding*. Po wciśnięciu klawisza, litera znika z okienka. Moduł wzrokowy zauważa zmianę w obrazie w wyniku czego automatycznie spróbuje odświeżyć swoją wiedzę.

Visual re-encoding

Świadomość istnienia tego mechanizmu jest ważna z dwóch powodów:

- nie możemy być nigdy pewni, że w międzyczasie zawartość bufora *Visual* się nie zmieni,
- nie możemy być nigdy pewni, że jeśli nie kierowaliśmy do niego zapytań to moduł wzrokowy będzie w stanie **free**.