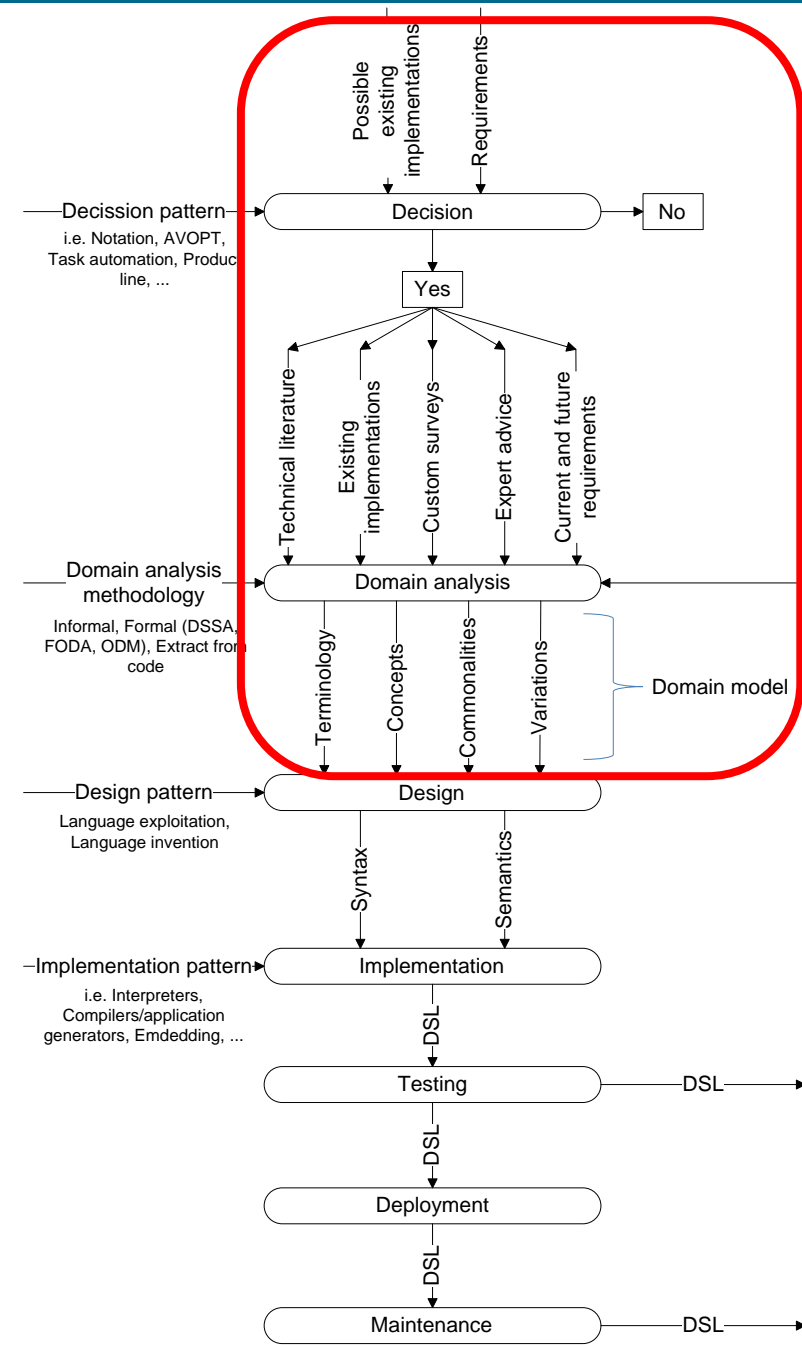


DOMENSKO SPECIFIČNI MODELIRNI JEZIKI 2024/2025

3. predavanje

Življenjski cikel



Faza odločitve za DSL

■ Vzorci odločitev

- Primeri kjer je razvoj DSLja smislen

■ Vaje

- Naredi **svoj jezik** za generiranje ogrodja projekta za sklad MERN
 - Vzorec odločitve „**Notacija**“
 - Ponuditi želimo prijazno notacijo
 - Vzorec odločitve „**Avtomatizacij nalog**“
 - Odpraviti ponavljajoče naloge (komponente)
 - Vzorec odločitve „**Produktne linije**“
 - Generiranje mase variant ene aplikacij
- V osnovi bomo **implementirali translacijo** iz svoje notacije v notacijo jezika JavaScript

Domenska analiza za DSL

- Vhod v domensko analizo
 - MERN program (naloga 1.2.)
- Definicija domene
 - Podani DSL **programi** (naloga 1.1.), **diagrami lastnosti** (naloga 2.1)
- Terminologija
 - **Besede in besedne zveze**, ki jih domenski **strokovnjaki** uporabljajo
 - Npr. components, api, render, ...
- Koncepti
 - **Niz konceptov** sestavlja jezik (analogija z naravnim jezikom: glagol, samostalnik, pridevnik, itd.)
 - Npr. spremenljivke, kontrolne strukture, podatkovne strukture, sintaksa, itd.
- Skupne lastnosti
 - **Skriti deli domene**, programerju nevidni klici, itd. (pozicioniranje posameznih delov modela)
- Spremenljive lastnosti
 - **Lastnosti** zapisane v programih (nav menu: yes / no)



Domenska analiza za DSL – primeri programov

■ Jezik CarModelLang (nalog a 1.1.)

```
car "MyCar1"  
  carbody SUV  
  color red  
  wheels 195/60R16
```

```
car "MyCar2"  
  carbody Limo  
  color white  
  roof black  
  roof rails silver  
  wheels 195/60R16
```

```
car "MyCar3"  
  carbody Minivan  
  color white  
  sunroof
```



Domenska analiza za DSL – terminologija

■ Jezik CarModelLang

```
car "MyCar1"  
  carbody SUV  
  color red  
  wheels 195/60R16
```

```
car "MyCar2"  
  carbody Limo  
  color white  
  roof black  
  roof rails silver  
  wheels 195/60R16
```

```
car "MyCar3"  
  carbody Minivan  
  color white  
  sunroof
```

Domenska analiza za DSL – koncepti

■ Jezik CarModelLang

■ sintaksa

```
car "MyCar1"  
  carbody SUV  
  color red  
  wheels 195/60R16
```

```
car "MyCar2"  
  carbody Limo  
  color white  
  roof black  
  roof rails silver  
  wheels 195/60R16
```

```
car "MyCar3"  
  carbody Minivan  
  color white  
  sunroof
```



Domenska analiza za DSL – skupne lastnosti

■ Jezik CarModelLang

■ sintaksa

```
car "MyCar1"  
  carbody SUV  
  color red  
  wheels 195/60R16
```

```
car "MyCar2"  
  carbody Limo  
  color white  
  roof black  
  roof rails silver  
  wheels 195/60R16
```

```
car "MyCar3"  
  carbody Minivan  
  color white  
  sunroof
```




Domenska analiza za DSL – spremenljive lastnosti

■ Jezik CarModelLang

■ sintaksa

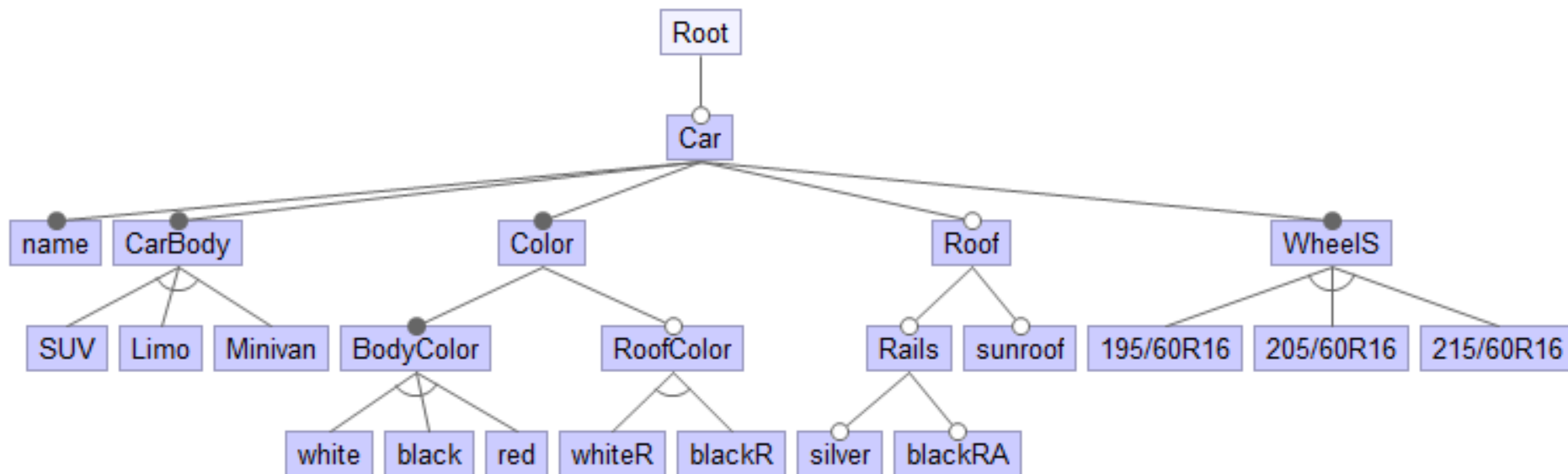
```
car "MyCar1"  
  carbody SUV  
  color red  
  wheels 195/60R16
```

```
car "MyCar2"  
  carbody Limo  
  color white  
  roof black  
  roof rails silver  
  wheels 195/60R16
```

```
car "MyCar3"  
  carbody Minivan  
  color white  
  sunroof
```

Domenska analiza za DSL – diagram lastnosti

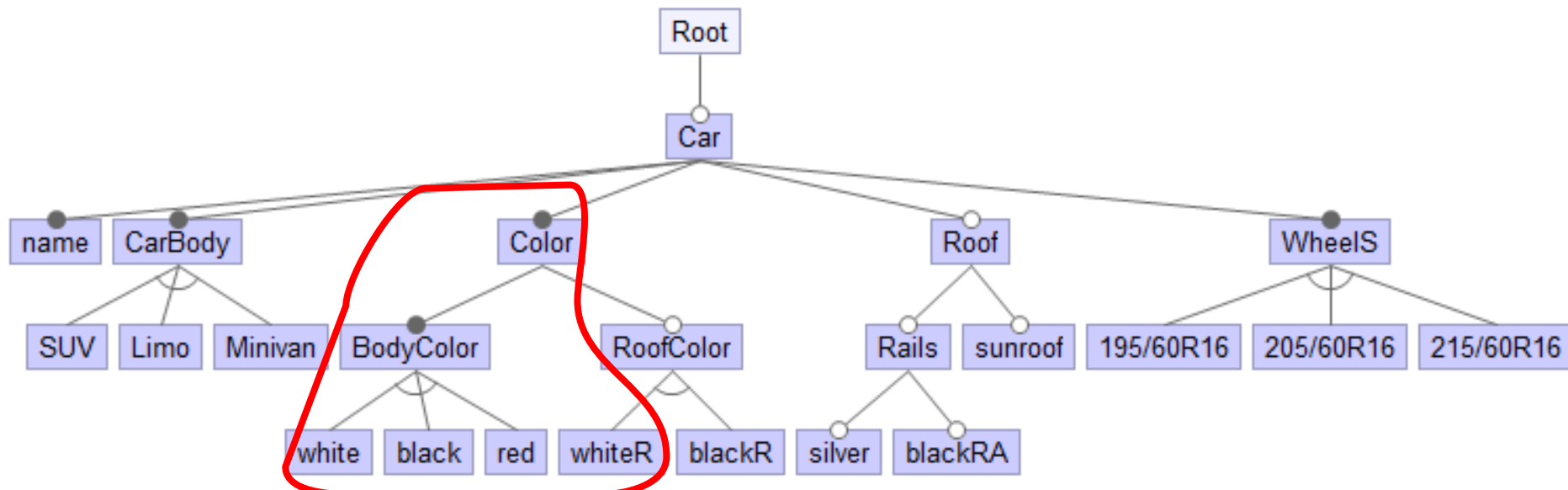
■ Jezik CarModelLang



Domenska analiza v diagram lastnosti

■ CarModelLang

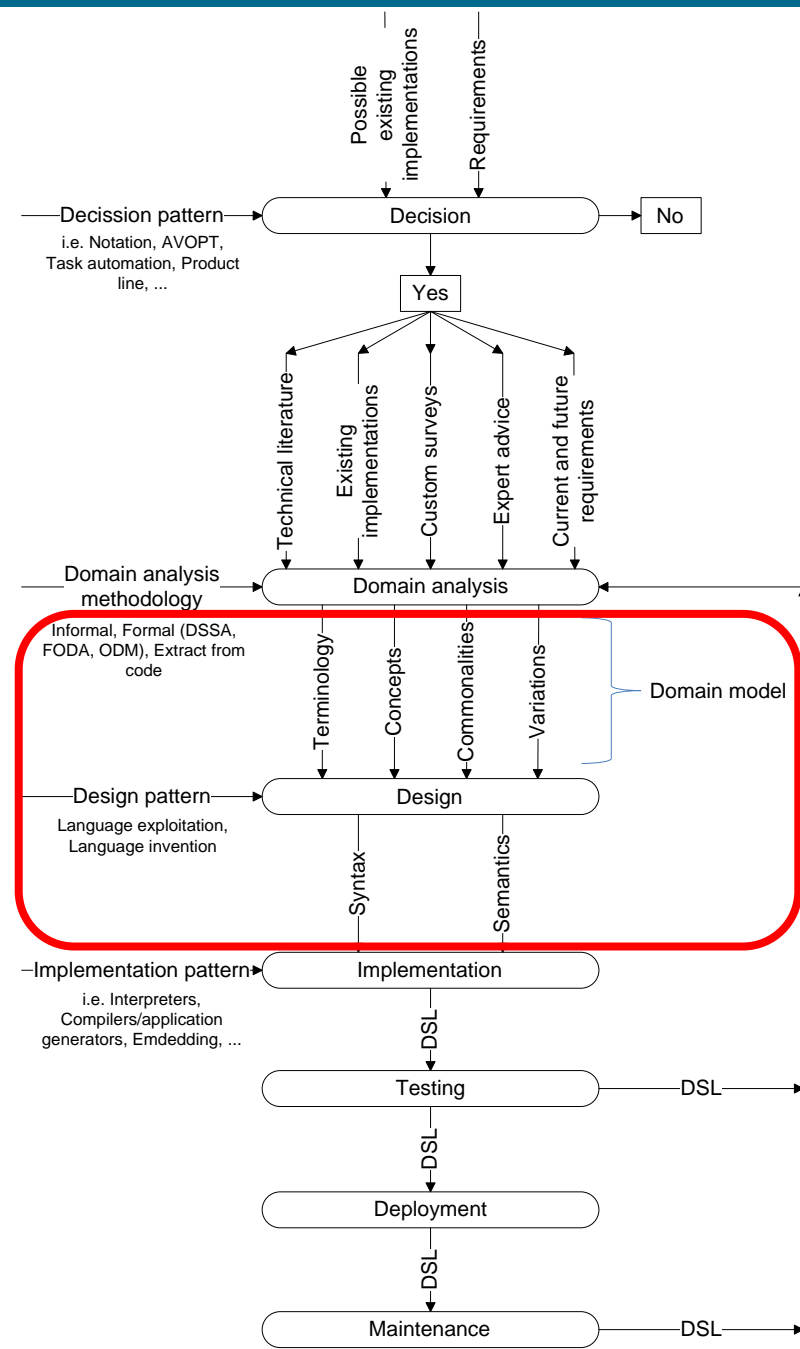
- Skupne lastnosti
 - Npr. Color, BodyColor
- Spremenljive lastnosti
 - white, black...



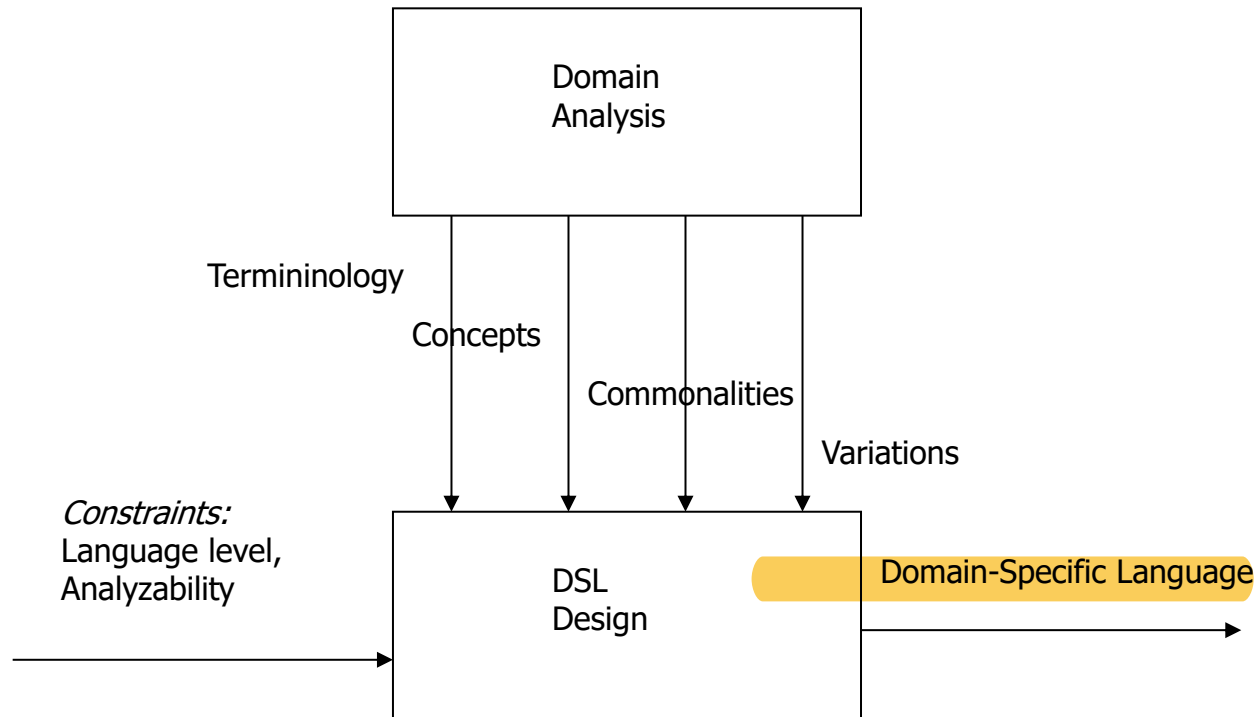
Domenska analiza DSLjev (ponovitev)

- **Seznam variacij** (spremenljivih delov domene) natančno prikazuje, katere informacije so potrebne za določitev primerka sistema. Te podatke je treba **neposredno določiti** ali jih **pridobiti** iz programov v DSLju.
- **Terminologija** in **koncepti** se uporabljajo za usmerjanje razvoja DSL konstruktov, ki ustrezajo variacijam.
- **Skupne značilnosti domene** se uporabljajo za opredelitev izvedbenega modela (angl. execution model) z množico skupnih operacij in primitivov jezika.

Življenjski cikel



Proces načrtovanja DSLjev



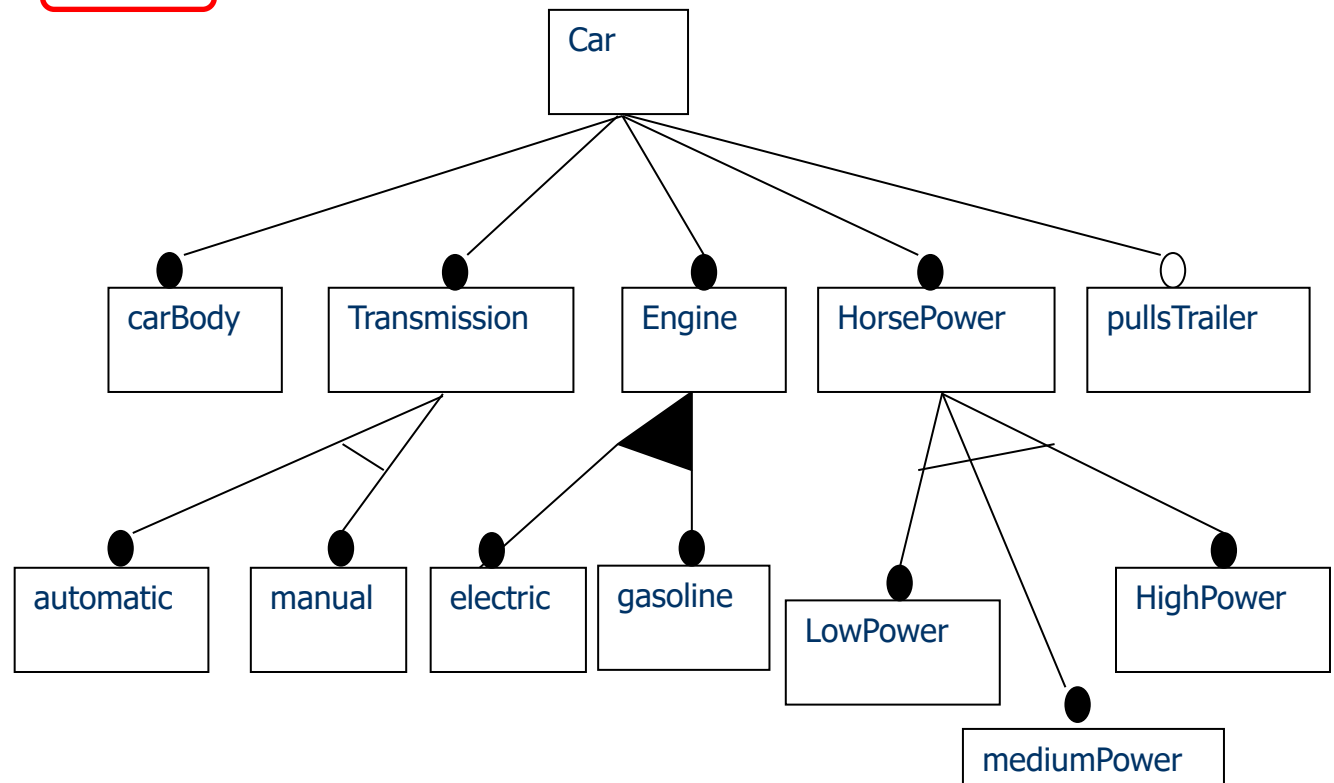
Cilji načrtovanja DSLjev

- Načrtovanje jezika vključuje opredelitev **konstruktov v jeziku** in **definiranje semantike jezika**, formalno ali neformalno.
- Semantika jezika opisuje **pomen** vsakega **konstrukta v jeziku**, ter fiksno obnašanje, ki ga ne pišemo v programu.
 - Izvedbeni model DSLja je veliko bogatejši kot v GPLjih.



Primer programa 1

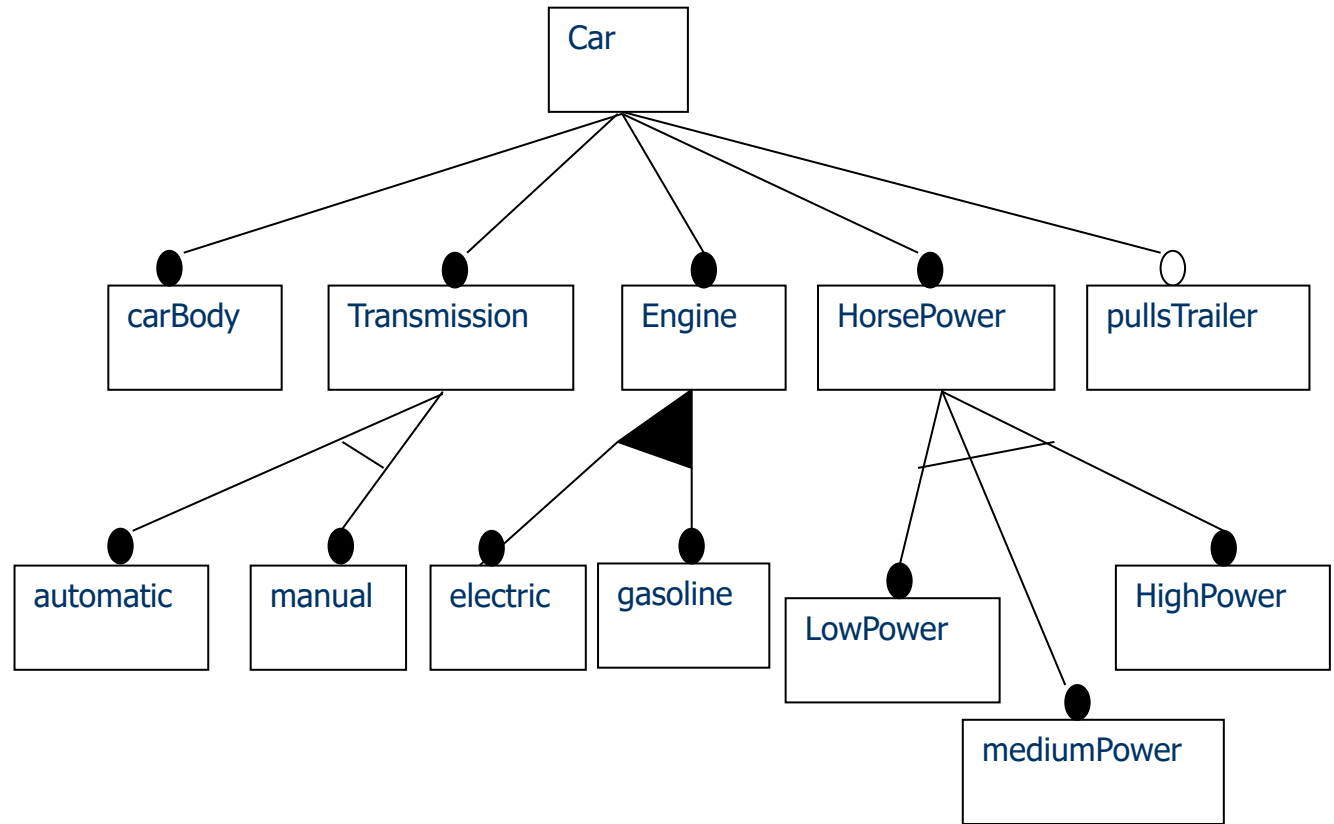
- Car: **all**(carBody, Transmission, Engine, HorsePower, pullsTrailer?)
- Transmission: **one-of**(automatic, manual)
- Engine: **more-of**(electric, gasoline)
- HorsePower: **one-of**(lowPower, mediumPower, highPower)





Primer programa 2

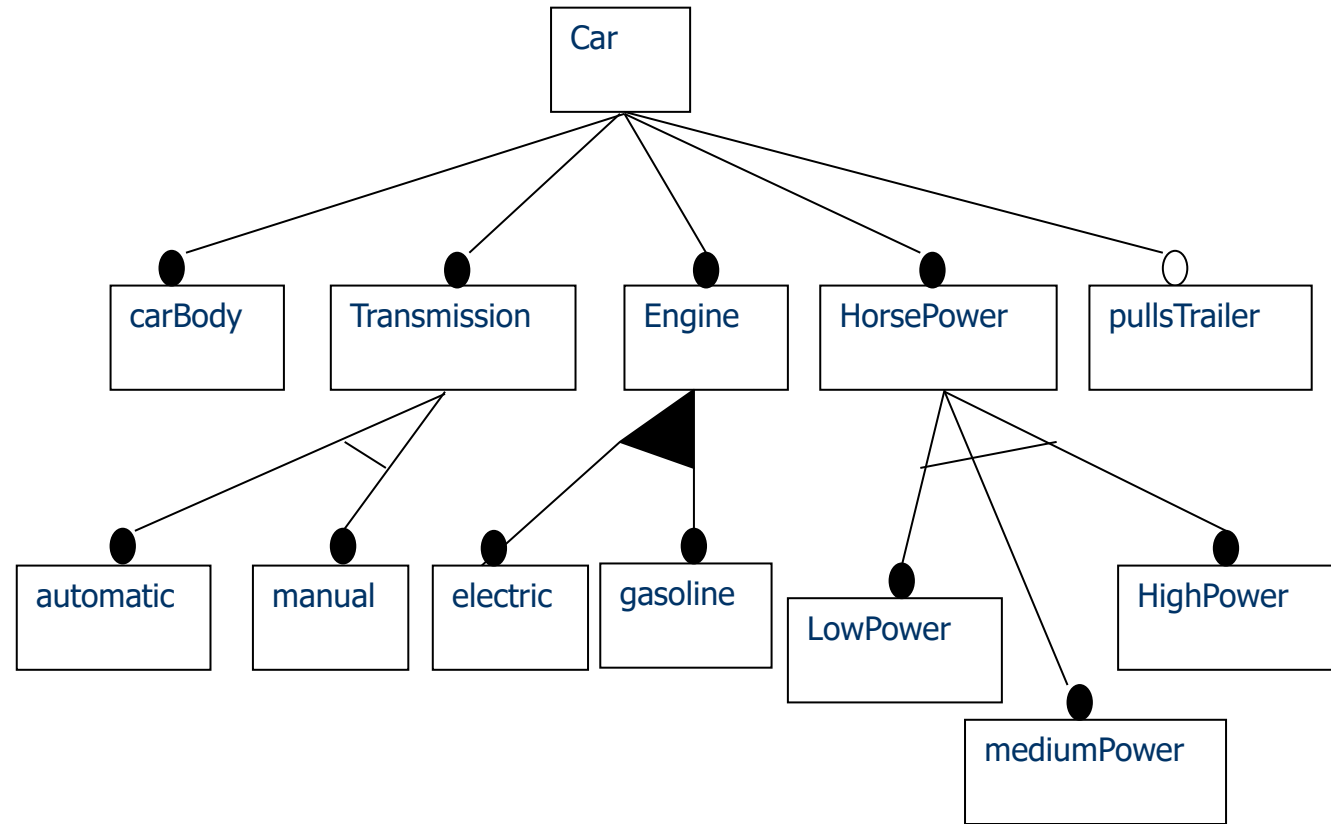
car (carBody,
transmission (automatic | manual),
engine (electric + gasoline),
horserPower (lowPower | mediumPower | highPower),
trailer [pullsTrailer])





Primer programa 3

car (carBody, transmission, engine, horsePower, trailer)
transmission (automatic | manual)
engine (electric + gasoline)
horsePower (lowPower | mediumPower | highPower)
trailer [pullsTrailer]



Oblikujemo lahko različne sintakse

- **Iz domenske analize** je mogoče razviti **različne jezike**, vendar vsi si delijo pomembne informacije, ki jih najdemo v diagramu lastnosti.
 - Spremenljive lastnosti so izražene drugače.
- Kateri DSL je **boljši**?
 - **Preučiti** moramo **principe** načrtovanja programskih jezikov.

Načrtovanje DSL – merila za dober jezik

- Iz predhodnih **izkušenj** z načrtovanjem programskih jezikov (PJ) in **kriteriji za ocenjevanje PJ**, so raziskovalci razvili merila za dobro oblikovanje PJ:
 - **Berljivost** (kako enostavno je brati in razumeti programe)
 - **Zapisljivost** (kako enostavno je pisati programe)
 - **Zanesljivost**
 - **Cena**
- Kategorije stroškov:
 - **Usposabljanje** programerjev
 - **Pisanje** programske opreme
 - **Sestavljanje**
 - **Izvajanje** programske kode
 - **Vzdrževanje**

Načrtovanje DSL - priporočilo

- Uporabljam **formalne metode** za določanje sintakse in semantike DSLja.
- Več prednosti:
 - **Sintaksa in semantika** se natančno in nedvoumno opredeli.
 - **Lažje dokazujemo lastnosti** programov in konstruktov
 - Edinstvena priložnost za **avtomatsko generiranje prevajalnikov** in interpreterjev
 - **Uporaba orodij** za načrtovanje programskega jezika
 - Programski jeziki, ki so bili zasnovani z eno od različnih formalnih metod, imajo boljšo sintakso in semantiko, manj izjem in se jih je lažje učiti.

Formalni pristopi za sintakso in semantiko

■ Programski jeziki

- Sintaksa
 - **BNF** in **EBNF** sta splošno sprejeta za formalen opis sintakse
- Semantika - **ni** nobene **standardne metode**; obstaja veliko pristopov: atributna gramatika, aksiomska semantika, operacijska semantika, denotacijska semantika, algebraična semantika, akcijska semantika, translacijska semantika, semantika na osnovi sledi

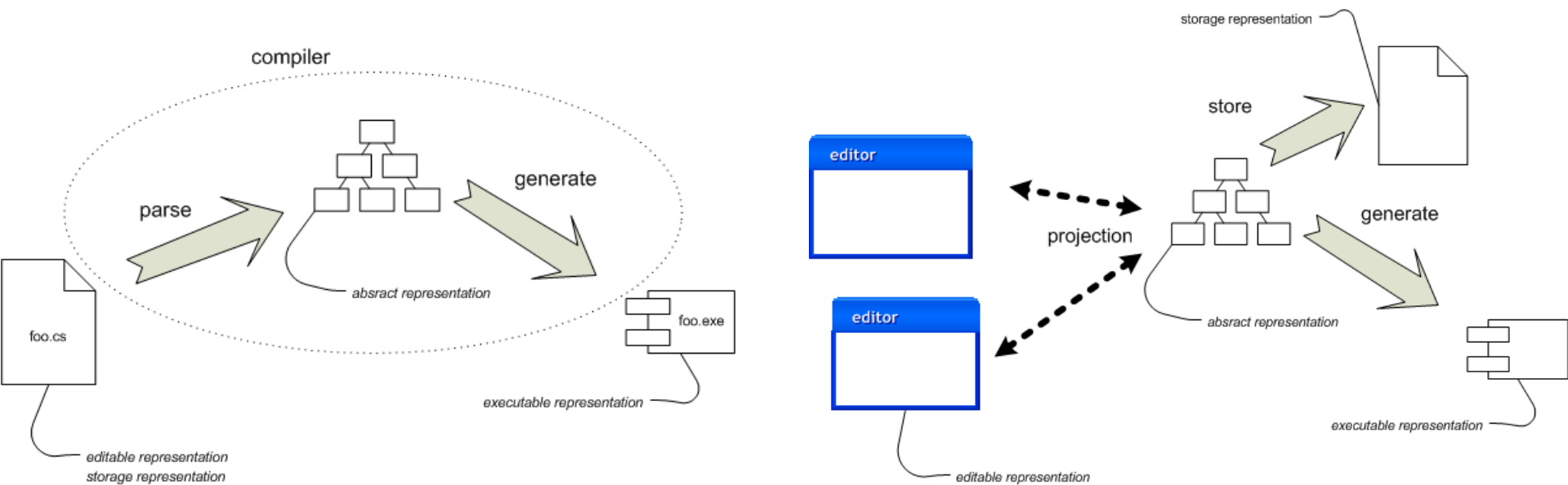
■ Projekcijski urejevalniki

- Sintaksa
 - Formalen zapis s pomočjo **AST** (angl. abstract syntax tree) in urejamo posamezna AST vozlišča (angl. AST nodes)
- Semantika
 - Generatorji kode: **transformacija**

■ Modeliranje jezikov

- Sintaksa - **metamodeli**
- Semantika - več pristopov (šablone, itd.)

Programskimi jeziki vs. projekcijski jeziki (urejevalniki)



Vir: [link](#)

Načrtovanje DSL – sintaksa z BNF

- **$G = (T, N, S, P)$** , $P \subseteq N \times (N \cup T)^*$, $S \in N$
 - T = terminal, N = netermina, S = startni terminal, P = produkcije
- **Produkcija** je naslednje oblike $A \rightarrow \alpha$; $\alpha \in (N \cup T)^*$
- Sintentična oblika γ G je katerikoli **niz terminalnih in neterminalnih simbolov**, ki jih lahko dobimo iz S .
 - γ_1 dobimo iz γ_2 ($\gamma_1 \Rightarrow \gamma_2$): if $\gamma_1 = \sigma\alpha\tau$ and
 - $\gamma_2 = \sigma\beta\tau$ and $\alpha \rightarrow \beta$
 - γ_1 dobimo iz γ_2 ($\gamma_1 \Rightarrow^* \gamma_2$): if $\gamma_1 \Rightarrow \sigma_1 \Rightarrow \dots \Rightarrow \sigma_n \Rightarrow \gamma_2$

Načrtovanje DSL – sintaksa z BNF

- **Kontekstno prost jezik** $L(G)$, izdelan iz gramatike G , je **set vseh nizov**, ki so **sestavljeni samo iz terminalnih simbolov**, ki jih lahko dobimo od začetnega simbola S s sekvenčno uporabo produkcijskih pravil.

- $L(G) = \{x \mid S \Rightarrow^* x \wedge x \in T^*\}$

- Primeri:

- Izrazi
- Robot



Načrtovanje DSL - gramatika

startni neterminal

Primer jezika izrazov:

produkcija $E \rightarrow T EE$ neterminal

$EE \rightarrow + T EE \mid \varepsilon$

$T \rightarrow F TT$

$TT \rightarrow * F TT \mid \varepsilon$

$F \rightarrow (E) \mid \text{\#int}$ terminal



Načrtovanje DSL – sintetična oblika

Primer jezika izrazov:

$$E \rightarrow T EE$$
$$EE \rightarrow + T EE \mid \varepsilon$$
$$T \rightarrow F TT$$
$$TT \rightarrow * F TT \mid \varepsilon$$
$$F \rightarrow (E) \mid \text{\#int}$$

Program: 3 * 5

$E \Rightarrow T EE \Rightarrow F TT EE \Rightarrow \text{\#int} TT EE \Rightarrow \text{\#int} * F TT EE \Rightarrow$
 $\text{\#int} * \text{\#int} TT EE \Rightarrow \text{\#int} * \text{\#int} EE \Rightarrow \text{\#int} * \text{\#int}$

sintetična oblika

Samo terminali → kontekstno prosta gramatika



Načrtovanje DSL – sintaktično drevo

Primer jezika izrazov:

$E \rightarrow T EE$

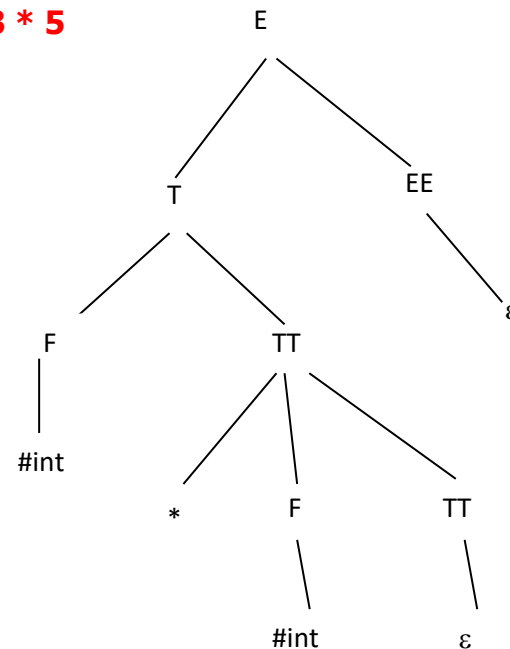
$EE \rightarrow + T EE \mid \varepsilon$

$T \rightarrow F TT$

$TT \rightarrow * F TT \mid \varepsilon$

$F \rightarrow (E) \mid \text{\#int}$

Program: 3 * 5





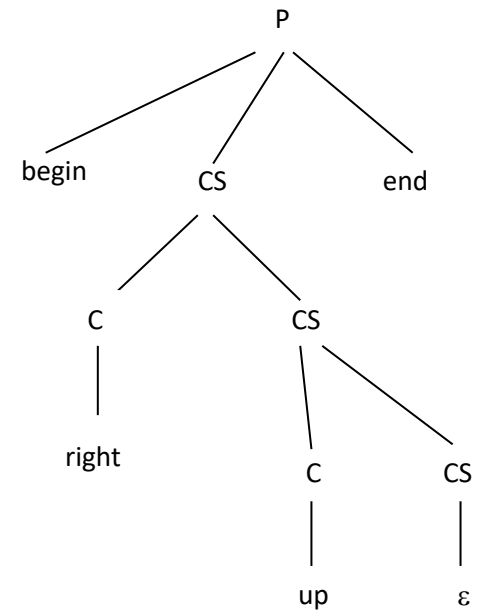
Načrtovanje DSL - sintaksa

Primer robota:

$P \rightarrow \text{begin CS end}$

$CS \rightarrow C CS \mid \varepsilon$

$C \rightarrow \text{left} \mid \text{right} \mid \text{up} \mid \text{down}$



Program: begin right up end

$P \Rightarrow \text{begin CS end} \Rightarrow \text{begin C CS end} \Rightarrow$
 $\text{begin right CS end} \Rightarrow \text{begin right C CS end} \Rightarrow$
 $\text{begin right up CS end} \Rightarrow \text{begin right up end}$

Načrtovanje DSL – sintaksa z EBNF

■ Razširitev BNF

■ Dodani simboli za:

- Opcija: ?
- Ponovitve
 - Nič ali več: *
 - En ali več: +

Primer robota BNF:

$P \rightarrow \text{begin } CS \text{ end}$

$CS \rightarrow C \ CS \mid \varepsilon$

$C \rightarrow \text{left} \mid \text{right} \mid \text{up} \mid \text{down}$



Primer robota v EBNF:

$P \rightarrow \text{begin } C^* \text{ end}$

$C \rightarrow \text{left} \mid \text{right} \mid \text{up} \mid \text{down}$

■ Prilagoditve:

- Ni epsilon (ε) produkcije

Načrtovanje DSL - sintaksa

- Ustvarjanje lastne sintakse jezika

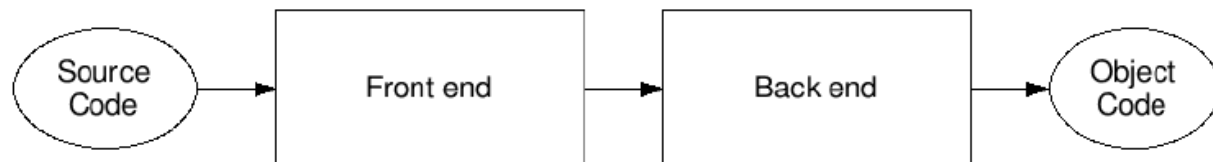
- Primeri:

- EGL
- MyArdoLang
- CarModelLang



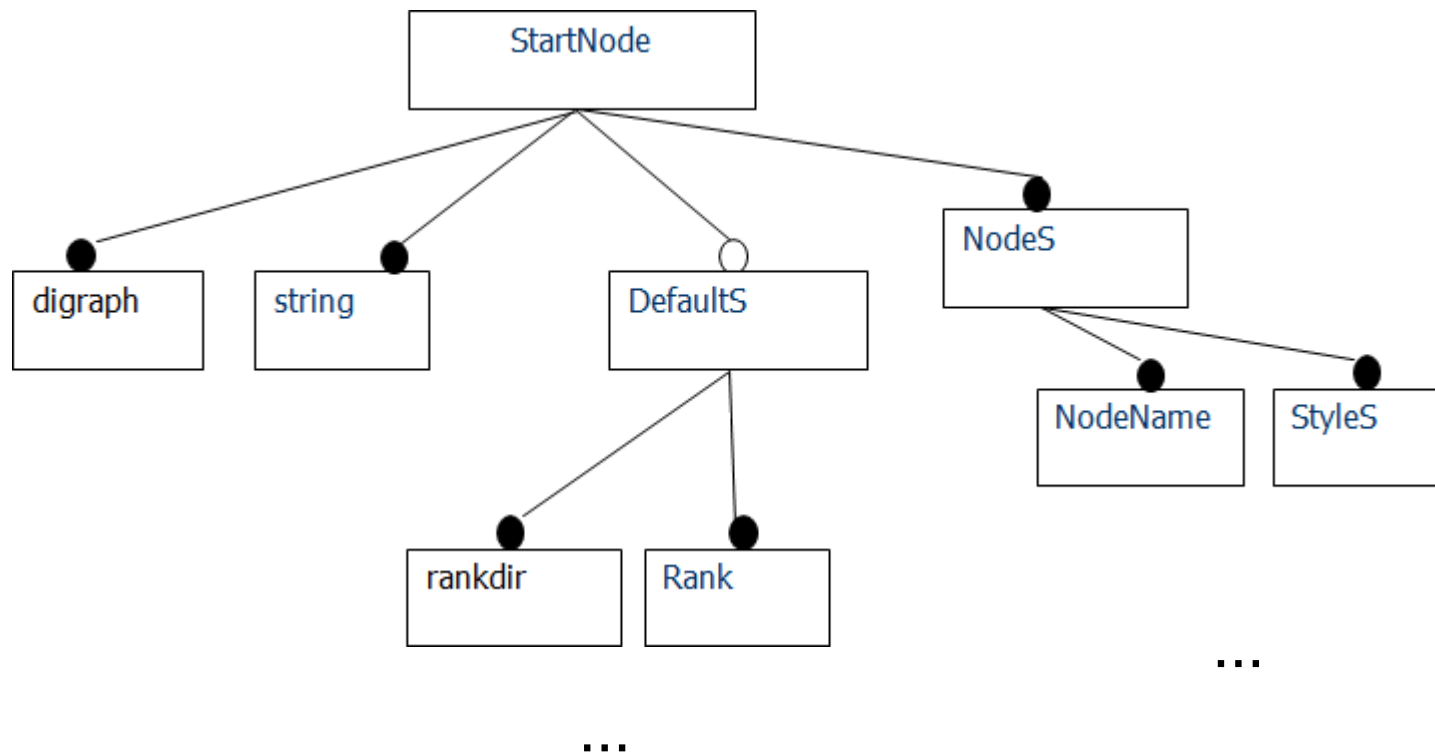
Primer EGL (EasyGraphLanguage)

```
digraph "Figure 2" {  
  rankdir=LR;  
  1 [label="Source\nCode"];  
  2 [label="Front end"  shape=box  
width=2.0 height=1.0];  
  3 [label="Back end"  shape=box  
width=2.0 height=1.0];  
  4 [label="Object\nCode"];  
  1 -> 2;  
  2 -> 3;  
  3 -> 4;  
}
```



Primer EGL (EasyGraphLanguage)

■ Domenska analiza





Primer EGL (EasyGraphLanguage)

■ Domensko načrtovanje

■ Gramatika – sintaksa

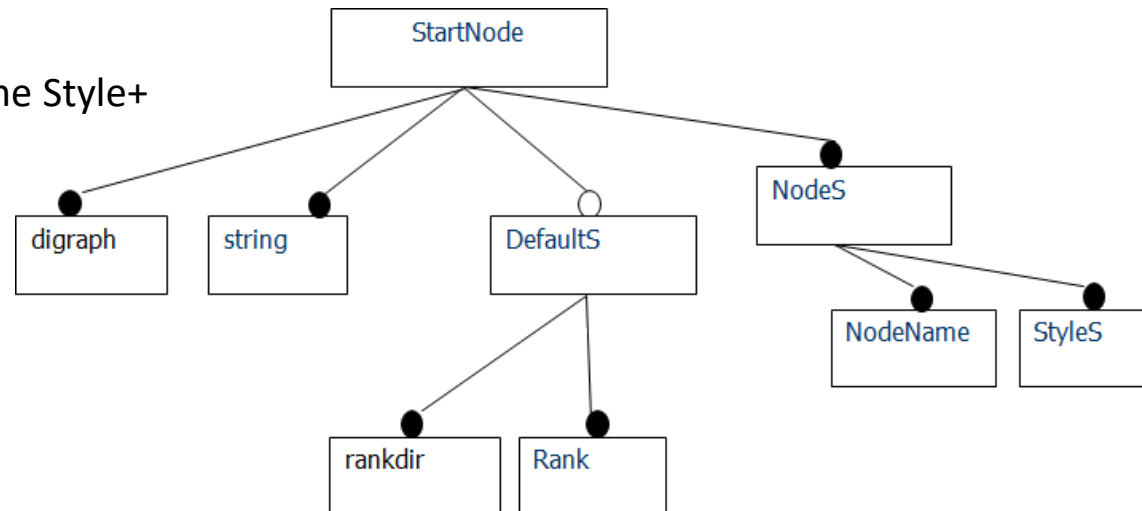
StartNode → digraph string Default* Node+

Default → rankdir Rank

Node → NodeName Style+

NodeName → INT

```
digraph Figure
    rankdir LR
    1 ...
    2 ...
```



```
digraph "Figure 2" {
    rankdir=LR;
    1 [label="Source\nCode"];
    2 [label="Front end" shape=box
    width=2.0 height=1.0];
    3 [label="Back end" shape=box
    width=2.0 height=1.0];
    4 [label="Object\nCode"];
```



Primer EGL (EasyGraphLanguage)

■ Domensko načrtovanje

■ Gramatika – dekoriranje sintakse

StartNode → 'digraph' STRING '{' Default* Node+ '}'

Default → 'rankdir' '=' Rank ';' // | OTHER

Rank → 'TD' | 'LR'

Node → NodeName '[' Style+ ']' ';'

NodeName → INT

Style → 'label' '=' STRING

 | 'shape' '=' Shape

 | 'style' '=' 'filled'

Shape → 'box' | 'oval'

```
digraph "Figure 2" {  
  rankdir=LR;  
  1 [label="Source\nCode"];  
  2 [label="Front end"  shape=box  
width=2.0 height=1.0];  
  3 [label="Back end"  shape=box  
width=2.0 height=1.0];  
  4 [label="Object\nCode"];
```

Enostavna pravila za pretvorbo iz diagrama lastnosti v sintakso

■ Domenska analiza in domensko načrtovanje

- **Uporaba enostavnih pravil za pretvorbo diagrama lastnosti v abstraktno sintakso**

- **Koncepti:**

- Obvezne lastnosti

- Sekvenca neterminalnih simbolov
 - $A \rightarrow B C D$

- Neobvezne lastnosti

- Sekvenca opsijskih neterminalnih simbolov
 - $A \rightarrow B? C? D?$

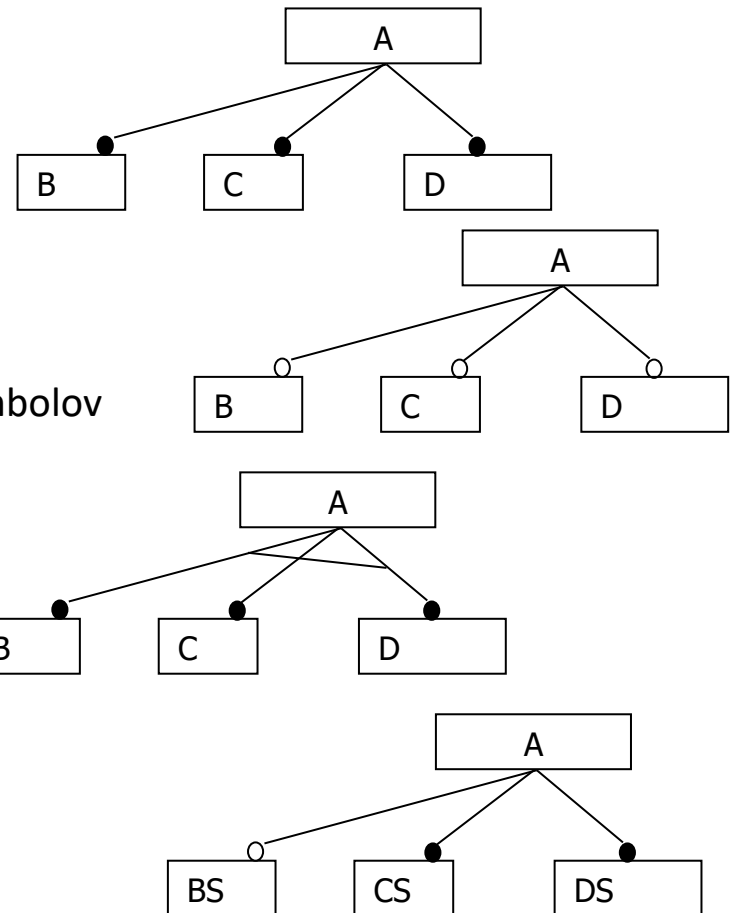
- Alternativne lastnosti

- Sekvenca neterminalnih simbolov
 - $A \rightarrow B | C | D$

- Kaj pa „Ali lastnosti“?

- Dodatno - ponovitve

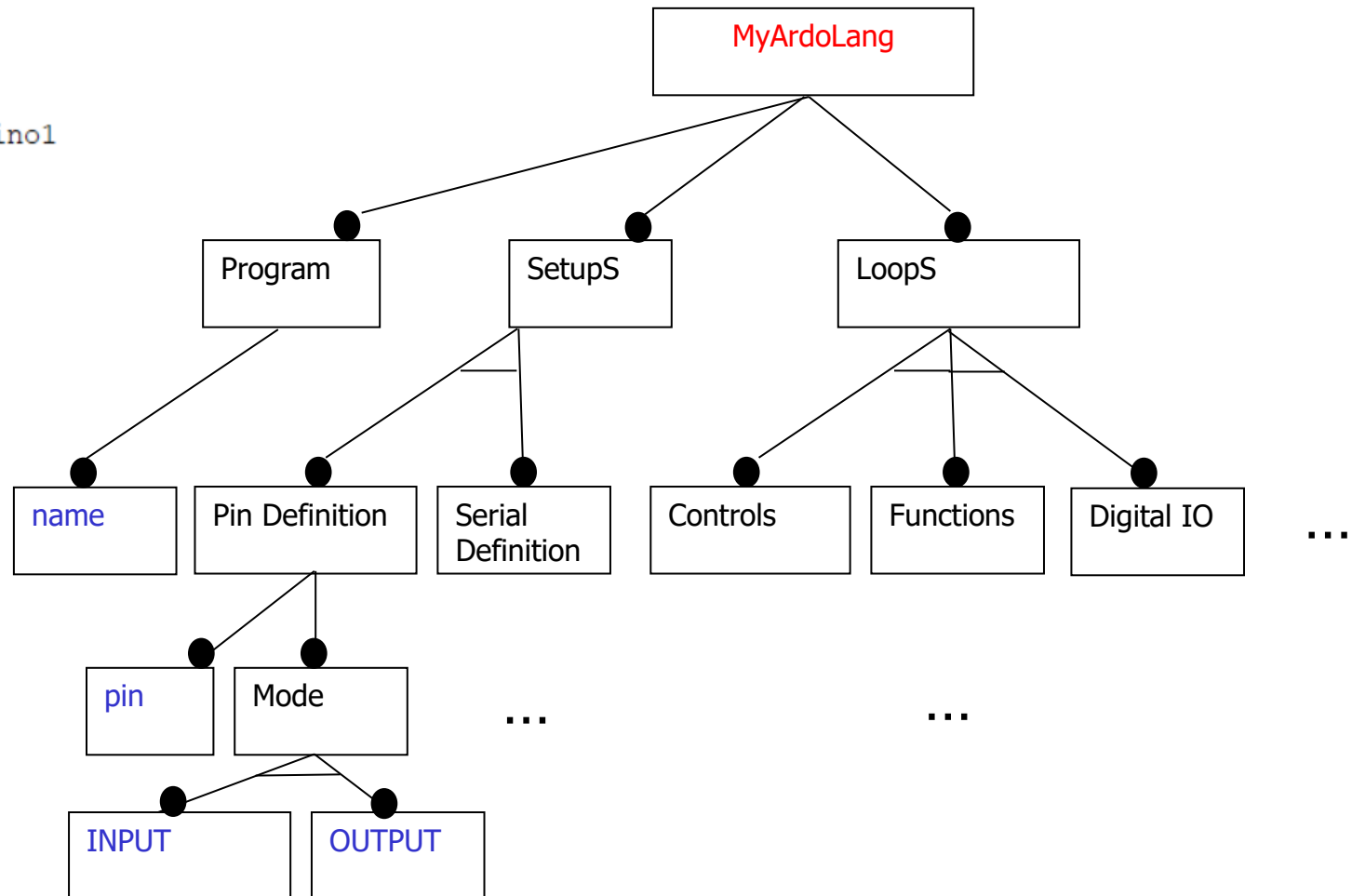
- $A \rightarrow B^* C^+ D^+$





Primer MyArdoLang

```
program EasyArduinol1
init
  D1 out
start
  D1 0
  delay 1000
  D1 1
  delay 2000
```





Primer MyArdoLang

■ 1. korak: osnovna sintaksa

MyArdoLang → Program Setup+ Loop+

Program → name

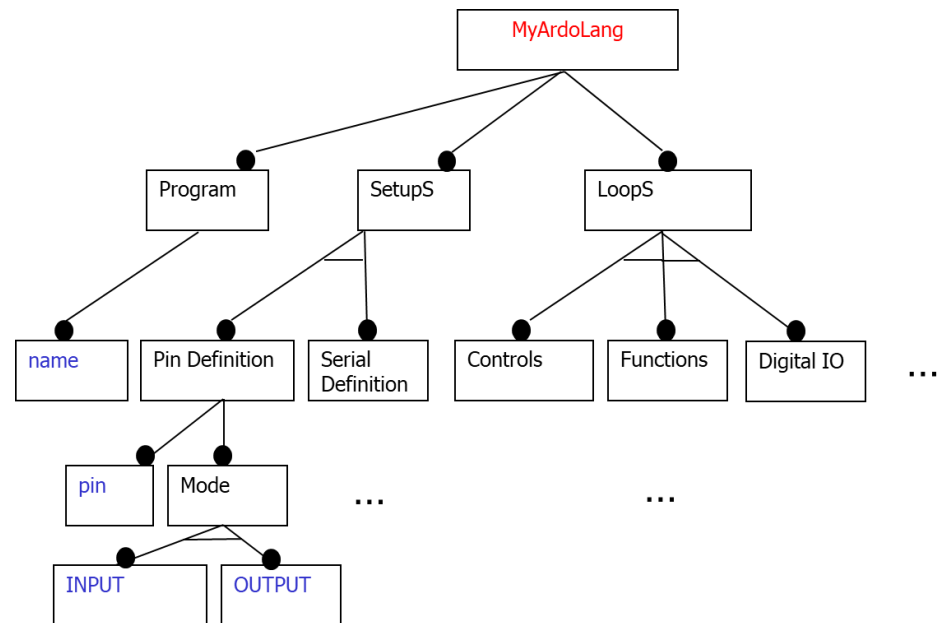
Setup → PinDefinition | SerialDefinition // | OTHERS

Loop → Control | Function | DigitalIO // | OTHERS

PinDefinition → pin Mode

Mode → INPUT | OUTPUT

```
program EasyArduinol
init
    D1 out
start
    D1 0
    delay 1000
    D1 1
    delay 2000
```





Primer MyArdoLang

■ 2. korak: dekoriranje sintakse

MyArdoLang → 'program' Program 'init' Setup+ 'start' Loop+

Program → name

Setup → PinDefinition | SerialDefinition // | OTHERS

Loop → Control | Function | DigitalIO // | OTHERS

PinDefinition → pin Mode

Mode → in | out

... // LOOP DEFINITION

```
program EasyArduinol
init
    D1 out
start
    D1 0
    delay 1000
    D1 1
    delay 2000
```



Primer CarModelLang

■ 1. korak: sintaksa

CarModelLang \rightarrow name CarBody Color Roof? Wheels

CarBody \rightarrow Suv | Limo | Minivan

Color \rightarrow BodyColor RoofColor?

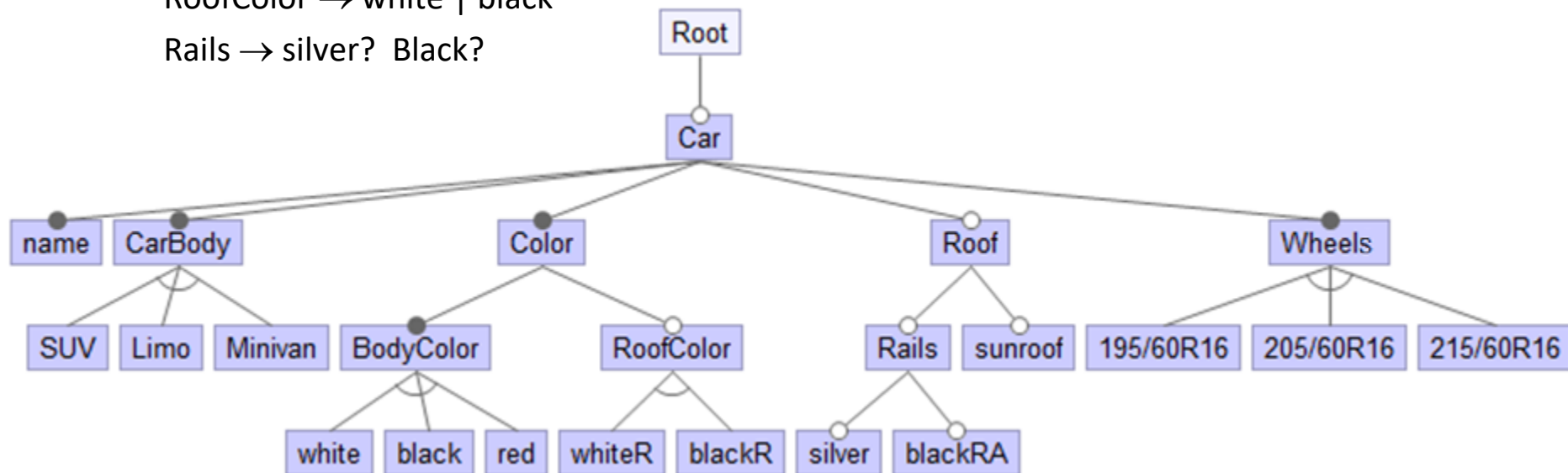
Roof \rightarrow Rails? sunroof?

Wheel \rightarrow 195/60R16 | 205/60R16 | 215/60R16

BodyColor \rightarrow white | black | red

RoofColor \rightarrow white | black

Rails \rightarrow silver? Black?





Primer CarModelLang

■ 2. korak: dekoriranje sintakse

CarModelLang \rightarrow 'car' name 'carBody' CarBody Color Roof? 'wheels' Wheels

CarBody \rightarrow 'suv' | 'Limo' | 'Minivan'

Color \rightarrow 'color' BodyColor ('roof' RoofColor)?

Roof \rightarrow ('roof' 'rails' Rails)? 'sunroof'?

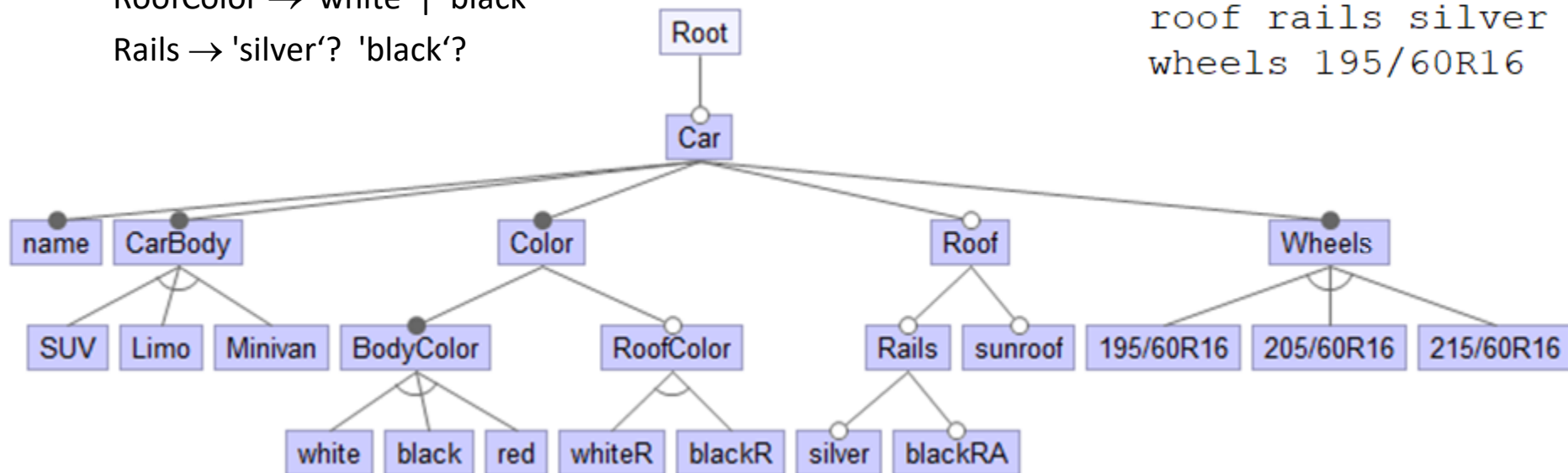
Wheels \rightarrow '195/60R16' | '205/60R16' | '215/60R16'

BodyColor \rightarrow 'white' | 'black' | 'red'

RoofColor \rightarrow 'white' | 'black'

Rails \rightarrow 'silver'? 'black'?

```
car "MyCar2"  
  carbody Limo  
  color white  
  roof black  
  roof rails silver  
  wheels 195/60R16
```



Načrtovanje DSL in programskih jezikov

- Načrtovalec DSLjev mora imeti v mislih tako posebnosti DSLjev kot tudi dejstvo, da uporabniki **niso** nujno **programerji**.
- GPLje so vedno oblikovali zelo **sposobni programerji**. Proces načrtovanja GPLjev je temeljil na njihovi **osebni estetiki** in **teoretičnih presojah**, kar je povzročilo izdelavo programskega jezika, ki ga oni **želili uporabljati**.

Načrtovanje DSL in programskih jezikov

■ Pri DSLjih drugače

- DSLji za končne uporabnike **ne smejo biti zasnovani s to intuicijo** saj načrtovalci DSLjev niso sami končni uporabniki.
- Načrtovanje DSLjev za končne uporabnike morajo voditi **empirične študije, vključitev končnih uporabnikov** in psihologijo raziskav programiranja.



Priporočila za načrtovanje DSLja

D. Wile. Lessons learned from real DSL experiments. Science of Computer Programming, Vol 51, Issue 3, pages 265-290, 2004.

- Message Type Sequence Language (MTSL)

action definitions:

definition hookup[ship] = wait 20 minutes

i.e., simulate hookup of a ship by waiting 20 minutes.

definition fuel[ship] = wait fuel-rate(ship) * fuel-room(ship);
assert fueled?(ship)

definition breakdown[ship] = wait 15 minutes; assert brokendown?(ship)

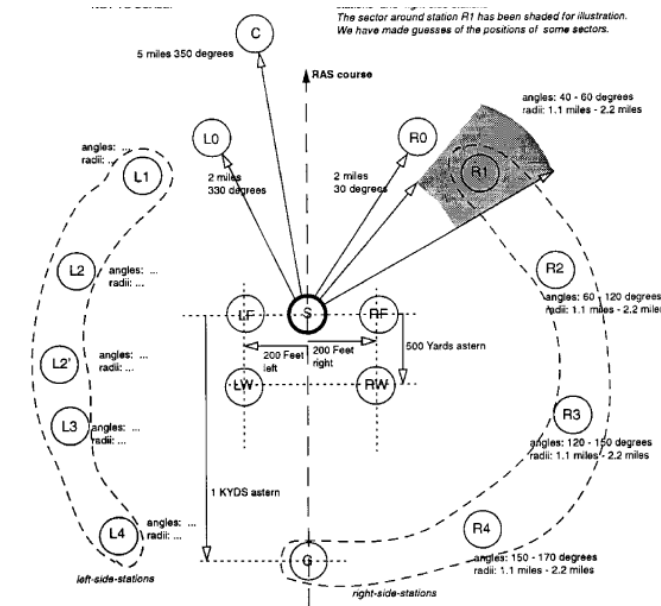
definition emergency-breakdown[ship] = wait 5 minutes; assert brokendown?(ship)

predicate definitions:

definition left-side-has-room? = not (every left-side-stations, s, is asg?s)

i.e., true if not every left side station has a ship assigned to it. Similarly,

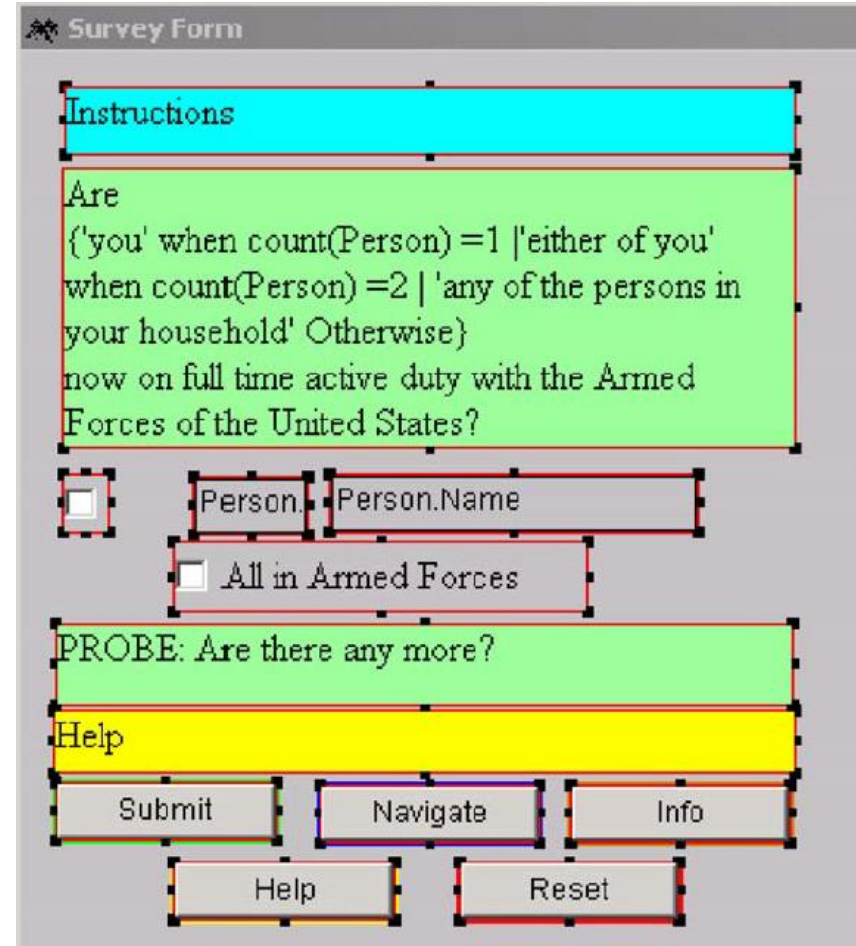
definition right-side-has-room? = not (every right-side-stations, s, is asg?s)



Priporočila za načrtovanje DSLja

- Survey instrument creator (SIC)

- Tehnološka vprašanja
- Organizacijska vprašanja
- Socialni problemi



The screenshot displays a 'Survey Form' interface with a DSL (Domain Specific Language) for defining survey questions. The DSL is written in a text area and uses curly braces for conditional logic based on the count of persons in a household.

Instructions

Are
{ 'you' when count(Person) =1 | 'either of you'
when count(Person) =2 | 'any of the persons in
your household' Otherwise}
now on full time active duty with the Armed
Forces of the United States?

☐ Person. Person.Name

☐ All in Armed Forces

PROBE: Are there any more?

Help

Submit Navigate Info

Help Reset

Tehnološka priporočila za načrtovanje DSLja

- Lekcija T1: Sprejmimo kakršne koli **formalne notacije**, ki jih **strokovnjaki domene že imajo**, namesto da izumljamo nove.
 - Lekcija T1 Predlog 1: uporabimo **njihove žargonske izraze** kadar koli je to mogoče.
 - Lekcija T1 Predlog 3: **Uporabimo splošno sprejete notacije**, ne izumljamo novih.
- Lekcija T2: Skoraj nikoli **ne načrtujemo programskega jezika**.
 - Lekcija T2 Predlog: **Načrtujemo** samo tisto, **kar je potrebno** (ne preveč). Naučimo se prepoznati svojo nagnjenost k prenačrtovanju.

Organizacijska priporočila za načrtovanje DSLja

- Lekcija O1: **Razumevanje organizacijskih vlog** ljudi, ki bodo uporabljali naš jezik.
 - Lekcija O1 Predlog 2: **Temeljito preučimo** trenutni **proces dela** preden ga nadomestimo z DSLjem.
- Lekcija O2: Prepričamo se, da je prenos našega izdelka (DSLja) v njihovo organizacijsko infrastrukturo, **skladen** z njihovim **poslovnim modelom**.

Socialna priporočila za načrtovanje DSLja

- Lekcija S1: Poiščimo zagovornika naše tehnologije v njihovi organizaciji.
 - Lekcija S1 Predlog: Vzpostavite **tesne vezi s strokovnjakom domene** za izdelavo infrastrukture v katero bo sistem prenesen.
- Lekcija S3: Ne pričakujmo od strokovnjakov domene, da vedo, kaj lahko računalnik naredi zanje.
 - Lekcija S3 Predlog 1: **Ne pričakujmo od strokovnjakov domene**, da bi razumeli, kaj računalnik ne more storiti zanje!
 - Lekcija S3 Predlog 2: Ne pričakujmo, da bodo uporabniki **prezrli ali razumeli naše napake v načrtovanju**.

Na kaj moramo misliti pri načrtovanju DSLjev?

■ Načela za načrtovanje DSLjev (Kolovos et al. 2006):

- **Skladnost** - kako DSL ustreza domenskim konceptom.
- **Ortogonalnost** - načelo znano iz načrtovanja GPLjev (sprememba v kodi povzroči spremembno na enem mestu in ne na več)
- **Podpora** - kako bo DSL podprt z nepogrešljivimi orodji (npr. urejevalniki, razhroščevanjem, testnimi ogrodji).
- **Integracija** - kako bo DSL integriran v druge procese oz. programsko opremo.
- **Dolgoživost** - DSL naj bi se uporabljal dovolj dolgo, da se bo razvoj DSLja in pripadajočih orodij izplačal.
- **Enostavnost** - načelo znano iz načrtovanja GPLjev (enostavnost izražanja)
- **Kakovost** - kako bo DSL podpiral kakovost in zanesljivost programske opreme.

Nasveti za oblikovanje jezikov (Horowitz)

- Izberite določeno **aplikacijo** (organizacija, poslovna logika, itd.)
- Načrtovalska skupina naj bo **majhna**
- Izberite natančne **cilje načrtovanja**
- Produkcijske **verzije načrta jezika predstavimo** majhni skupini zainteresiranih ljudi
- **Revidiramo** (popravimo/dopolnimo) programski jezik
- **Zgradimo prevajalnik** in napisati formalne definicije jezika - **semantiko**
- Izdelamo jasne in natančne **priročnike**
- Zagotovimo „**produkcijsko kakovost**“ prevajalnika in široko distribucijo
- Napišite **primere**, ki opisujejo jezik

Nasveti za oblikovanje jezikov

- Ne vključujemo **nepreizkušenih idej** - konsolidacija, ne inovacija.
- **Preprostostost je ključ** - izogibajmo se zapletenosti. Preveč možnosti otežuje razumevanje jezika. (načelo KISS)
- Izogibajmo se **zapletenim zahtevam**, da se nekaj pove več kot enkrat.
- **Avtomatiziramo** mehanske, dolgočasne in **aktivnosti**, ki so nagnjene k napakam (zagotovimo funkcionalnosti višjega nivoja).
- Izogibajmo se funkcionalnostim, ki so **odvisne** od določenega naprave ali majhnega števila različnih naprav.
- **Pravila brez izjem**, se je lažje naučiti, uporabljati, opisati in implementirati.
- Programski jeziki so za ljudi - oblikovanje jezikov za človeka pomeni **povečanje produktivnosti**.

Kompromisi pri načrtovanju DSLjev

- Zanesljivost vs. stroški izvajanja
- Berljivost vs. zapisljivost
- Fleksibilnost vs. varnost

- Ali se načrtovanje DSLjev zelo razlikuje od načrtovanja GPLjev?
 - Mnogi raziskovalci verjamejo, da se načrtovanje DSLjev **ne razlikuje** veliko od načrtovanja GPLjev.

Odprte težave DSL načrtovanja

- Kako **analiza domene** vpliva na proces načrtovanja jezika?
- Kdo naj **načrtuje DSL**? Domenski strokovnjak, načrtovalec GPLja ali programski jezikovni inženir?
- **Koliko analize domene in načrtovanja** je dejansko potrebno?
- Ali lahko **preskočimo nekaj** začetnih **razvojnih faz** DSLja? Kakšne so posledice?
- Ali lahko **razvijamo podporna orodja** že v zgodnejših fazah razvoja DSL?

Naslednja naloga na vajah (naloga 3.1)

- Za DSL iz diagrama lastnosti (naloge 2.1) zapišite EBNF. DSLju morajo ustrezati primeri programov iz naloge 1 s katerimi testiramo gramatiko.

```
CarModelLang → 'car' name 'carBody' CarBody Color
Roof 'wheels' Wheels
CarBody → 'suv' | 'Limo' | 'Minivan'
Color → 'color' BodyColor ('roof' RoofColor)?
Roof → ('roof' 'rails' Rails)? 'sunroof'?
Wheels → '195/60R16' | '205/60R16' | '215/60R16'
BodyColor → 'white' | 'black' | 'red'
RoofColor → 'white' | 'black'
Rails → 'silver' | 'black'
```

Naslednje vaje

■ Vaje ta teden

- 1. skupina (RV 1): torek, 13.00 – 14.30, F-201
 - 2. skupina (RV 2): torek, 09.00 – 11.30, G-219
 - 3. skupina (RV 3): sreda, 12.00 – 13.30, Lumiere
 - 4. skupina (RV 4): sredo, 18.00 – 19.30, Lumiere
-
- 1. skupina (RV 1): četrtek, 09.30 – 11.00, F-102
 - 2. skupina (RV 2): četrtek, 11.00 – 12.30, F-102
 - 3. skupina (RV 3): petek, 08.00 – 09.30, F-201
 - 4. skupina (RV 4): petek, 09.30 – 11.00, F-201

Vprašanja

