2. ŠPECIFIKÁCIA DIGITÁLNYCH SYSTÉMOV

A. Koncepcia špecifikácie
B. Agent
C. Proces
D. Špecifikácia časovania (rozšírenie agenta)
E. Štandardný zápis špecifikácie s agentami a procesmi

A. Koncepcia špecifikácie

Pod špecifikáciou systémov rozumieme (formálny) opis správania alebo štruktúry a prípadne iných vlastností a parametrov systému. Opis systému zvonka_bez opisu jeho vnútornej štruktúry sa nazýva behavioristický opis, alebo **behavioristická špecifikácia** – funkcie. Ide o **opis správania** – funkcie, teda ide o to **čo systém robí** vo vnútri vzhľadom na svoje rozhranie s okolím, a nie ako to robí vo svojej vnútornej štruktúre. Systém môže byť opísaný aj j **štruktúrnou špecifikáciou**, teda špecifikáciou opisujúcou **štruktúru systému**. Vzhľadom nato, že správanie všetkých prvkov štruktúry ako aj ich vzájomné prepojenie sú dané, behavioristická špecifikácia je tu implicitne definovaná. Transformácia zo štruktúrnej špecifikácie na behavioristickú je netriviálna úloha. Nazýva sa **analýza** Opačná transformácia z behavioristickej na zodpovedajúcu štruktúrnu špecifikáciu sa nazýva **syntéza**.

Špecifikácia systému by mala byť z hľadiska súčasných potrieb návrhu opísaná v niektorom **formálnom jazyku**. V systémoch automatizovaného návrhu na RTL a logickej úrovni návrhu sa najčastejšie používajú jazyky VHDL a Verilog. Na systémovej úrovni sa používa viacero jazykov SystemC, Ocapi, Esterel, SpecCharts a iné. Nazývajú s systémové návrhové jayky (Systém Design Languages SDL, alebo System specification and design L. - SSDL). Sú koncipované pre systémovú úroveň a sú pre túto úroveň spravidla vhodnejšie.

V ďalšom, aby sme charakterizovali **najdôležitejšie potreby špecifikácie**, pri opise správania modulov (subsystémov) digitálneho systému, ktoré majú byť implementované HW spôsobom, budeme používať opis správania vychádzajúci z vlastného modelu spracovania informácie a to jednak na systémovej a tiež na RTL úrovni,. Na tomto modeli predstavuje bázu formálneho behavioristického špecifikačného jazyka HSSL, ktorý bol spracovaný na FIIT. V podstate opisuje systém spriahnutých, vzájomne komunikujúcich konečných stavových strojoch (automatov). Konečný stavový stroj (FSM – Finite State Machine) je v tomto modeli a v jazyku HSSL vyjadrujeme entitou, ktorú nazývame agent (agent správania). **Agent** opisuje, alebo ako tiež hovoríme, vykonáva - exekuje – nejaké (zmysluplné aj zložité) **čiastkové správanie** v danom digitálnom systéme (subsytéme) v **konečnom časovom intervale**. Spriahnutie (kompozíciu) agentov (teda viacerých FSM) opisujeme v našom modeli a v jazyku HSSL prostredníctvom **procesov**.

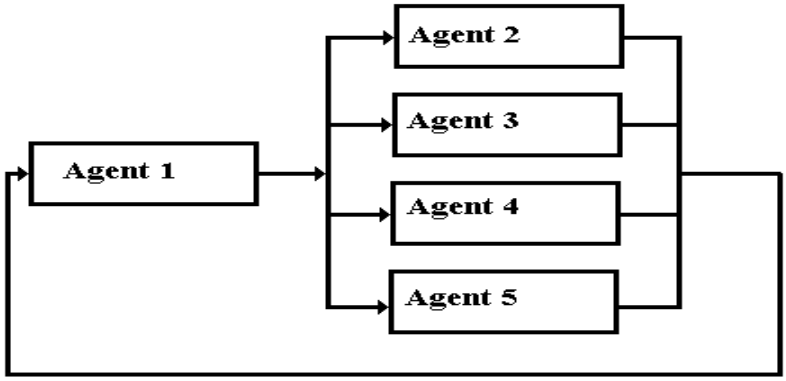
Výhodou tohto prístupu je, že môžeme názorne vysvetliť **zjemnenie** (vývoj) **špecifikácie** z najvyššej systémovej úrovne až do úrovne RTL, ktorá je pri návrhu našou cieľovou úrovňou. Na cvičeniach, pri konkrétnych prípadových štúdiách (príkladoch) sa budeme zaoberať opisom a **behavioristickou** a **štruktúrnou špecifikáciou** navrhovaného digitálneho

systému na RTL úrovni prostriedkami jazyka **VHDL**, ktorého znalosť tam predpokladáme.

Podstatu spomenutej koncepcie používanej v prednáškach ilustrujú nasledujúce príklady.

PRÍKLAD 1: Predstavme si, že v správaní niektorého číslicového systému môžeme vyčleniť 5 charakteristických čiastkových správaní, každé v niektorom konečnom časovom intervale, a tieto správania pripíšeme pôsobeniu (vykonávaniu) určitých "agentov" (Agent-1,…, Agent-5) v systéme. Čiastkové správania systému vlastné jednotlivým agentom si možno predstaviť opísané napr. pomocou **konečných stavových strojov** (**automatov**). Potom správanie systému (ako celku) možno vyjadriť určitým **spriahnutím** agentov napr. tak, ako to naznačuje nasledujúci obrázok.

Špecifikácia (opis správania) daného systému opísané **procesom**, ktorý vytvára určité **spriahutie** agentov je znázornené na obrázku.



Chápeme (interpretujeme) to takto: Pri štarte systému, ako prvý, začne pôsobiť (sa začne vykonávať v sytéme) Agent-1, t.j. realizuje sa správanie vlastné tomuto agentu. Po skončení pôsobnosti sa spustí jeden z vybraných agentov (ako **alternatíva**) Agent-j (j ≥ {2,..,5}). Po skončení pôsobnosti agenta Agent-j sa proces opakuje, t.j. zaradí sa na vykonanie znova Agent-1 atď. Výber agenta sa riadi podľa určitej logickej podmienky, predikátu, ktorý je definovný nad premennými systému. Spriahnutie agentov – ktoré nazývame **proces,** sme tu vyjadrili grafom.

Práve opísaná behavioristická špecifikácia môže zodpovedať špecifikácii aktívneho procesora CPU z príkladu v kapitole 1, pričom jednotlivým agentom priradíme nasledujúce čiastkové správania (funkcie):

Agent-1: Vybrať inštrukciu z externej pamäti RAM a jej prenesenie do RI;

IR := M[PC] a do PC zapísať adresu nasledujúcej inštrukcie

Agent-2: Vykonanie inštrukcie ADD (účinok: AC := AC + M[adresa])

Agent-3: Vykonanie inštrukcie LD (čítanie RAM a presun údaja do AC;
AC := M[adresa])

Agent-4: Vykonanie inštrukcie ST (presun obsahu AC do RAM;
M[adresa] := AC)

Agent-5: Vykonanie inštrukcie BRN (skok na danú adresu, ak číslo
v AC je záporné; teda ak najvyšší bit AC je 1, potom
PC := adresa, inak
PC:= PC + 1

"adresa" reprezentuje konkrétnu hodnotu zložky "adresa" inštrukcie, t.j. obsahu <IR.adresa>.

Podmienka alternatívneho výberu je hodnota (obsah) zložky "opkod" inštrukcie, t.j. <IR. opkod>, napr. pri <IR. opkod> = ADD sa vyberie Agent-2.

Čiastkové správanie vlastné agentovi Agent-1 je možno opísať aj nasledujúcim **Mealyho stavovým strojom** (automatom) **M1=** (DV, S1, DH, pr1, vs1), kde DV je množina **vstupných vektorov** {(<D>,<Res>,<Wait>} systému CPU, DH je množina **výstupných vektorov** {<D>, <A>, <RD/WR>, <Req>} CPU, S1 je **množina stavov** {RES, IF0, IF1, IF2, OD}; pr1, vs1 sú **prechodová** a **výstupná** funkcia, ktoré sú zapísané v (riadkovej) prechodovej tabuľke

Stavy sú dané nasledujúcimi vektormi hodnôt stavových premenných (pozri príklad CPU v kap.1)

(<MBR>, <MAR>, <PC>, <IR>, <AC>, <Contr>),

kde Contr je stavová premenná, ktorá určuje tzv. "stav riadenia" Premenná Contr nadobúda v danom prípade hodnoty z množiny {res,if0, if1, if2, od}. Každá z týchto premenných nadobúda hodnotu 1 práve iba len vtedy, ak je systém v príslušnom (rovnomennom) stave.

Prechodová tabuľka automatu M1 opisujúca čiastkové správanie Agent-1

Stav	**Hodnoty vstupov ;**	**Hodnoty výstupov**	**Nasleduj úci stav**
IF0	**Wait = 0 ;**		**IF0**
IF0	**Wait = 1 ;**	**RD/WR=1, Req=1, A=PC**	**IF1**
IF1	**Wait = 1 ;**	**RD/WR=1, Req=1, A=PC**	**IF1**
IF1	**Wait = 0, D=d; ;**	**Req = 0**	**IF2**
IF2	**Wait = 0 ;**	**u**	**IF2**
IF2	**Wait = 1 ;**	**u**	**OD**
OD	**u ;**	**u**	**- -**

S1= {**IF0**,...,**OD**} sú tzv. globálne stavy automatu M1 dané nasledujúcimi vektormi hodnôt

(<MBR>, <MAR>, <PC>, <IR>, <AC>, <Contr>).
IF0 = (u, u, a‘, u, u, if0)
IF1 = (u, u, u, u, u, if1), **IF2** = (u, u, u, u, u, if2)
OD = (d, u, d.adresa, d.opkod, u, od) -- OD je skratka pre Operation Decode

Stav **IF0** je **začiatočný** daný hodnotou <PC> =a‘. Hodnota „a’ “ v PC je adresa inštrukcie v externej pamäti, ktorá sa má čítať pri správaní opísanom Agent-1 z externej pamäti. Pri **koncovom** stave **OD** je v systéme CPU stav daný obsahmi registrov, teda hodnotami stavovými premenných <IR> = d.opkode, <PC> = d.adresa, a hodnotou Contr = od. Konkrétna hodnota “d” v IR je hodnota načítaná zo vstupného portu D, interpretovaná ako „opkod“ inštrukcie; d.opkod є{LD, ST, ADD, BRN}. Symboly „u“ v jednotlivých stavoch interpretujeme z hľadiska špecifikácie ako **nerelevantné** (nešpecifikované hodnoty stavových premenných).

V stavoch sú teda špecifikované iba tie hodnoty stavových premenných MAR,…,AC, ktoré sú **relevantné** pre opis čiastkového správania systému agentom. Udávajú sa iba na **začiatku** (začiatočný stav) a najčastejšie iba na **konci** (v koncovom stave) vykonávania agenta.

Obsahy registrov – hodnoty príslušných stavových premenných MBR,…,AC na konci pôsobenia agenta, ktoré agent **nastavuje**, môžeme skrátene zapísať dobre známymi priraďovacími príkazmi:

IR := D, PC:= PC+ 1 (MBR := D), nemení sa <MAR> = a‘, <AC> =u
Hodnota na **ľavej** strane priraďovacieho príkazu (:=) je na **konci** exekúcie agenta a hodnota na **pravej** strane, na jej **začiatku** exekúcie.

Pri východiskovej - hrubej, menej detailnej - špecifikácií CPU možno vypustiť stavové premenné MAR a MBR, ktoré sme uviedli v danom príklade v kap.1. Pre opis funkcie na danej úrovni abstrakcie nie sú nevyhnutné, postačuje zaviesť PC a IR.
Poznámka: Prechodová tabuľka M1 je zapísaná v tzv. **riadkovej forme**. Riadky sú priradené dvojiciam: (stav, V/V vektor). Tabuľku interpretujeme takto: Napr., ak je stav IF0 a vstupný vektor je taký, v ktorom je <Wait>=1, potom vo výstupnom vektore <RD/WR>=1, <Req>=1 a <A>=<PC> a nasledujúci stav je IF1.

Správanie zodpovedajúce agentovi Agent-5 môžme opísať napr. Mealyho nedeterministickým stavovým strojom **M5** = (DV, S5. DH, p5, v5), ktorý je v ďalšej tabuľke.

Stav	**Hodnoty Vstupov ; hodnoty výstupov**	**Nasledujúci stav**
OD s obsahom d.opkod = BRN	**u ; u**	**BR0, BR0‘**
BR0 s obsahom <AC> < 0, (c=1)	**u ; u**	**IF0**
BR0’ s obsahom <AC> >= 0, (c=0)	**u ; u**	**IFO**
IF0	**u ; u**	**--**

V začiatočnom stave **OD** stavového stroja **M5** je <IR> = **BRN**, „opkod“ tejto inštrukcie.

Stavy sú reprezentované vektormi hodnôt stavových premenných (<MBR>, <MAR>, <PC>, <IR>, <AC>, Contr): **OD** = ((u, u, d.opkod, d.adresa, a, br0).

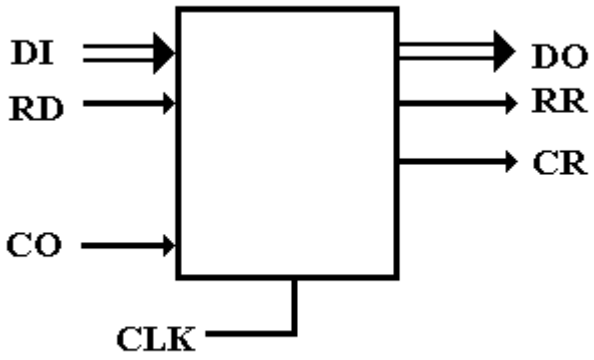
Zo stavu **OD** je prechod do oboch stavov **BR0** a **BR0'** pri každom vstupnom vektore. Ide teda o **nederministický** stavový stroj; pri u je prechod do oboch stavov. Tento stroj možno však prepísať na deterministický (Prechod do stavu **BR0** alebo **BR0'** zo stavu **OD** odlíšime alternatívne pomocou hodnoty stavov premennej AC, <AC>=a, nastavenej v predchádzajúcom inštrukčnom cykle a teda hodnotou predikátu **c** ako súčasťou vstupného vektora.)

V tabuľke sú **BR0** a **BR0' vnútorne** odlíšené predikátom **c** závislom od obsahu „a" AC. Predikát **c** = 1 ak <AC>= a < 0, alebo **c** = 0 ak <AC> >= 0. Zápis d.adresa v PC znači adresu; v danom prípade adresu skoku d.adresa pri rozvetvení a d.opkod v IR značí kód inštrukcie BRN vybratej z pamäti, ktorá sa má vykonať v tomto prípade v CPU.

V koncovom stave **IF0** agenta Agent-5 platí: Contr = if0. Ak pri začiatočnom stave **OD** je <AC> < 0 (c=1), tak v koncovom stave **IF0** je PC:= d.adresa, inak PC := PC + 1.

Treba si všimnúť, že stavy v oboch uvedených prechodových tabuľkách stavových strojov často vyjadrujú vlastne relatívne mohutné **podmnožiny množiny globálnych stavov,** pričom konkrétny stav v tejto podmnožine je daný konkrétnymi hodnotami <PC> = d.adresa, <IR> = d.opkod, <AC> => c, Contr = contr (teda vybratou konkrétnou inštrukciou („d") z pamäti, adresou „d.adresa" v „d", predikátom „c", ktorý je nositeľom informácie o obsahu AC, a napokon, konkrétnym aktuálnym riadiacim stavom contr. Stavy v tabuľke predstavujú vlastne určité **množiny stavov**, tzv. **zlučiteľných** stavov. Tak napr., podmnožina BR0 je daná všetkými stavmi (u, u, u, <AC>=0, Contr =if0), ktoré sú **zlučiteľné**, t.j. vo vektore hodnôt sú na všetkých miestach **rovnaké**, kde sú (konkrétne) špecifikované.

PRÍKLAD 2: Vezmime pasívny procesor VSYS z príkladu v kapitole 1.



V danej koncepcii bude systém špecifikovaný iba množinou dvoch agentov, ktoré nazveme ReadIn a Compute, pričom daným agentom je vlastné takéto správanie:
ReadIn: Z okolia sa prenesie pole 16 čísel $d_1,..,d_{16}$ (cez D) do vnútornej pamäti RAM

(vykoná sa to sekvenčne, číslo po čísle, systémom so vzájomným potvrdením - handshaking, pomocou signálov REQ a ACK. Agent končí svoje pôsobenie nastavením signalizačného výstupu (Read Ready) RR=1.

Compute: Postupne číta 16 čísel uložených vo vnútornej pamäti RAM a akumulatívne sa vypočíta ich súčet. Agent končí nastavením signalizačného výstupu (Comute Ready) **CR**=1. Pritom súčet čísel tvorí hodnotu stavovej premennej **DO** a objaví sa aj v dátovom výstupe **DO**.

Na tomto príklade uvedieme spôsob, akým sa "štartuje“ vykonávanie agentov a procesov“, t.j. ako sa spustí čiastkové spávanie vlastné agentovi, resp. opísané procesom (hovoríme jednoducho o "štarte alebo naštartovaní" agenta resp. procesu)

V danom systéme VSYS v špecifikácií sú iba dvaja agenti a **štartujú** sa takto:

ReadIn: Ak sa objaví udalosť up(CLK=1) a ak **RD**=1

Compute: Ak sa objaví up(CLK=1) a ak **CO**=1

RD a **CO** sú logické vstupy VSYS (inštrukčné signály zvonku).

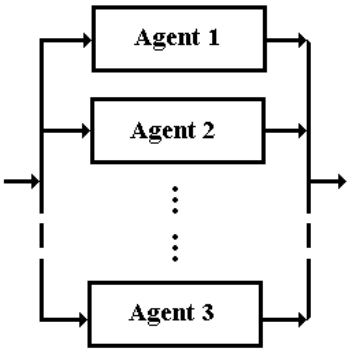
Takýto systém nazývame **štartovací mechanizmus** (vysvetlíme ho podrobne neskoršie.

Podobný **systém** ako VSYS, ktorý namiesto vstupov **RD** a **CO** má iný vstup **START**, možno opísať aj pomocou uvedených dvoch agentov a nasledujúceho **procesu PR**:

Štartovací mechanizmus procesu spustí tento proces ak je splnená táto podmienka
up(CLK=1) a **START**=1 a nebeží PR

POZNÁMKA: V príkladoch sme videli len dva s možných typov spriahnutia agentov a to **sekvenčné spriahnutie**, nazývané aj **zreťazenie** a alternatívne spriahnutie, ktoré nazývame **alternatívou** (v danom prípade podmieneným rozvetvením). Používa sa aj **paralelné** (**súbežné** alebo konkurenčné) spriahnutie, pozri obr.,ktoré nazývame aj paralelizmom n agentov.

Pri



tejto kompozícií sa naštartujú naraz všetci agenti a proces sa ukončí ak všetci skončili.

Algebra agentov *(hrubý náčrt)*

Pre danú množinu MA agentov možno formálne zaviesť nasledujúcu algebru agentov:

(MA, O, NA),

kde

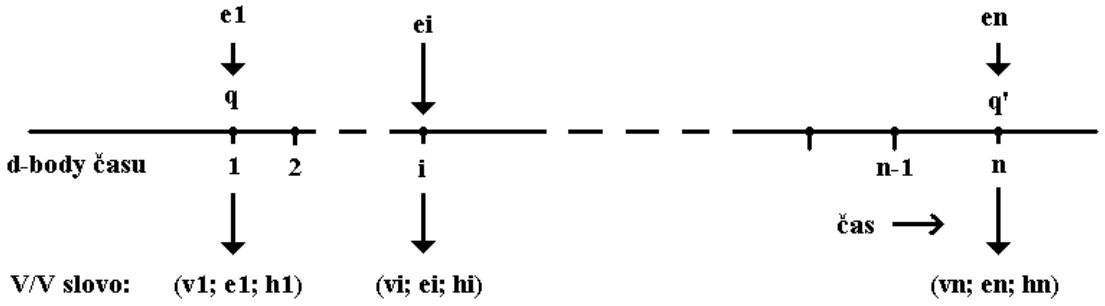
1. NA je **nulový** agent (pozri ďalej)
2. O je množina **operácií** definovaných na agentoch, obsahujúca
"." zreťazenie , "+" alternatívu a "||" paralelné spriahnutie, pre ktoré platí:

FA A ϵ MA: **NA.A = A.NA = A**
(FA – „For All“ značí logický všeobecný kvantifikátor "pre každý")
FA A,B,C ≥ MO: **A + B = B + A**
A.(B + C) = A.B + A.C
(B + C).A = B.A + C.A
A.(B || C) = A.B || A.

B. Agent

V tejto časti **formálne opíšeme** agenty (správania) tak, ako ich budeme používať pri špecifikácií zložitého systému. Namiesto opisu čiastkového správania systému konečným stavovým strojom použijeme iný spôsob opisu a to

- formálny opis komunikácie **komunikačnej množiny KM** systému (kapitola 1) pomocou **špeciálnych výrazov** (formúl),
- separáciu tzv. **operačných stavov** agenta
- zavedenie funkie (**g**) nastavujúcej operačný stav na konci exekúcie agenta, a teda aj hodnoty jednotlivých **operačných stavových** premenných Z_{i1}, Z_{i2}, …, Z_{ij} z množiny operačných premenných agenta.



KM je množina **všetkych možných** komunikačných slov (časovaných V/V slov), typu

(v1;e1;h1) (v2;e2;h2).....(vn;en;hn),

ktoré indikuje system a okolie pri čiastkovom správaní opisovanom (vykonávanom) agentom

Operačné stavy sú stavy **operačnej časti** (alebo operaačných častí) primárnej architektúry (resp. kompozície primárnych architektúr) navrhovaného digitálneho systému, ktorého čiastkové správanie daný agent špecifikuje.

Ak je daný **začiatočný** (globálny) **stav** „s“, potom **koncový** operačný stav „q“ pre aktuálne vstupné slovo $\underline{v}$ v komunikačnom slove „w“, ktoré systém indikoval, je daný spomenutou funkciou **g**

$$q = \mathbf{g}(s, \underline{v})$$

Uvedenými dvomi entitami **KM** a **g** je agent **kompletne** definovaný (z hľadiska opisu správania v diskrétnom čase).

Globálny stav tu chápeme ako množinu všetkých dvojíc stavov z množiny **R**, tzv. **riadiacich stavov** (stavov stavového stroja **KSM** vo formálnom modeli agenta, pozri ďalej) a už spomenutých operačných stavov z množiny **Q**, teda množina globálnych stavov je **S** = **R x Q** = { s = (r, q) | r ϵ **R**, q ϵ **Q** }.

V **príklade** Agent-1 v CPU sú tieto množiny takéto: **R** = {<Contr>} = { IR0, IR1, IR2, OD}, **Q** = { q | q = (<MBR>, <MAR>, <PC>, <IR>, <AC>) } a v danom konkrétnom prípade platí **S** = **RxQ** = { (<MBR>, <MAR>, <PC>, <IR>, <AC>, <Contr>).

Formálne modely agenta

Najskôr si všimneme kompletný **stavový model** agenta. Agent A digitálneho systému môžeme formálne opísať jeho formálnym stavovým modelom. Opíšeme ho štvoricou

$$A =_{def} (\mathbf{S}, \mathbf{SI}, \mathbf{KSM}, \mathbf{Q}, \mathbf{g})$$

SI je **množina začiatočných stavov**, ktorá je podmnožinou množiny **S** globálnych stavov systému **S** = **R** x **Q**

Q = { (<Z1>, …., <Zk>); <Zr > ϵ DZr, r = 1,2,…, k)} je komletná množina operačných stavov kde {Z1,Z2,…,Zk} je množina **operačných stavových premenných** systému označovanou symbolom Z a **R** je množina riadiacich stavov (pozri **KSM**)

KSM je (konečný stavový stroj- FSM), ktorý nazývame **riadiaci** (**komunikačný**) stavový stroj agenta. Je opísaný šesticou

$$\mathbf{KSM} = (\mathbf{SI}, DV, DH, \mathbf{R}, \mathbf{vs})$$

DV a DH, je množina vstupných, resp. výstupných vektorov systému. Sú tvorené vektormi hodnôt primárnych vstupných, resp. výstupných premenných (portov) systému.

R je množina **riadiacich stavov**, stavov **KSM** agenta A, pričom platí: **Q** ∩ **R** = Ø. Stav r ε **R** je reprezentovaný vektorom hodnôt určitých stavových premenných, ktoré v modeli neuvádzame.

vs je **výstupná sekvenčná** funkcia

vs : **SI** x DV***** → DH*****

pričom DV***** a DH***** sú množiny vstupných resp. výstupných **slov** (**konečných reťazcov**) zostavených s prvkov z množiny vstupných (DV) resp. výstupných (DH) vektorov systému,

Funkcia **vs** každému začiatočnemu globálnemu stavu „s " a každému vstupnému slovu v priraďuje (vracia) určité výstupné slovo h. Teda h = **vs**(q, v). Obidbe slova v a h sú obsiahnutému (zabalené) v **komunikačnom slove** w

Pretože predpokladáme, že **KSM** je **konečný** stavový stroj (Mealyho alebo Moorovho typu) výstupnú sekvenčnú funkciou **vs** (rozšírenu na vstupné a výstupné slová) pre KM možno zostaviť pomocou známej, iba na vstupné slová rozšírenej výstupnej funkcie FSM

vr : **SI** x DV***** → DH,

ktorá každému stavu „s" z **SI** a každému stupnému v slovu z DV***** priraďuje (vracia) po dosadení do **KSM** niektorý výstupný „h" (pozor!: nie výstupné slovo h), teda h = **vr**(q, v).

Pri dĺžke lg=1 vstupného slova v (teda jedden vtupný vektor) ide o **jednoduchú** výstupnú funkciu v : Rx DV → Dhstavového stroja. Pre **KSM** možno zostaviť aj tzv. **rozšírenú prechodovú funkciu pr**, ktorá stavu s ε **SI** a vtupnému slovu v priraďuje stav po skončení tohto slova.

pr : **SI** x DV***** → **R**

Pri dĺžke lg=1 vstupného slova (teda pri jednom vstupnom vektore) **pr** plní funkciu jednoduchej prechodovej fukcie

p: **R** x DV → **R**

Funkciu **vs** možno **vypočítať** (vyjadrť) pomocou funkcie **vr** takto:
Pre začiatočný stav stav s,vstupné slovo v = v1v2v3….v(n-1)vn

vs(s,v1v2v3….v(n-1)vn) = vr(s,v1) vr(s,v1v2) vr(s,v1v2v3)……vr(s, v1v2v3….vn) = h1h2h3……hn,

kde h1h2h3……hn je potom výstupné slovo h, ktoré vracia sekvenčná funkcia **vs** pre daný stav „s" a vstupné slovo v.

Výpočet **vs** a **vr** možno urobiť pomocou jednoduchých funkcií v **KSM**: prechodovej funkcie „p" a výstupnej funkcie „v".

Sekvenčná funkcia „**vs**" je **sekvenčným zobrazením**, ktoré

(1) zachováva dľžku (lg) slov (teda pre každý stav s ϵ **SI** a každé vstupné slovo r ϵ DV* platí lg(vs (q, r)) = lg(r)

(2) zachováva začiatočné úseky výstupných slov (teda platí: pre každý stav s ϵ **SI** a pre každé dve vstupné slová r1 = p. b1 (p je prefix slova r a b1 jeho pokračovanie) a r2 = p.b2 s rovnakým prefixom p platí:
vs (q,r1) = u.c1 **vs** (q,r2) = u. c2, kde u je ten istý prefix a lg(u) = lg(u)

Funkcia „**g**" agenta A je funkcia pre **nastavenie operačných stavov** z množiny **Q**

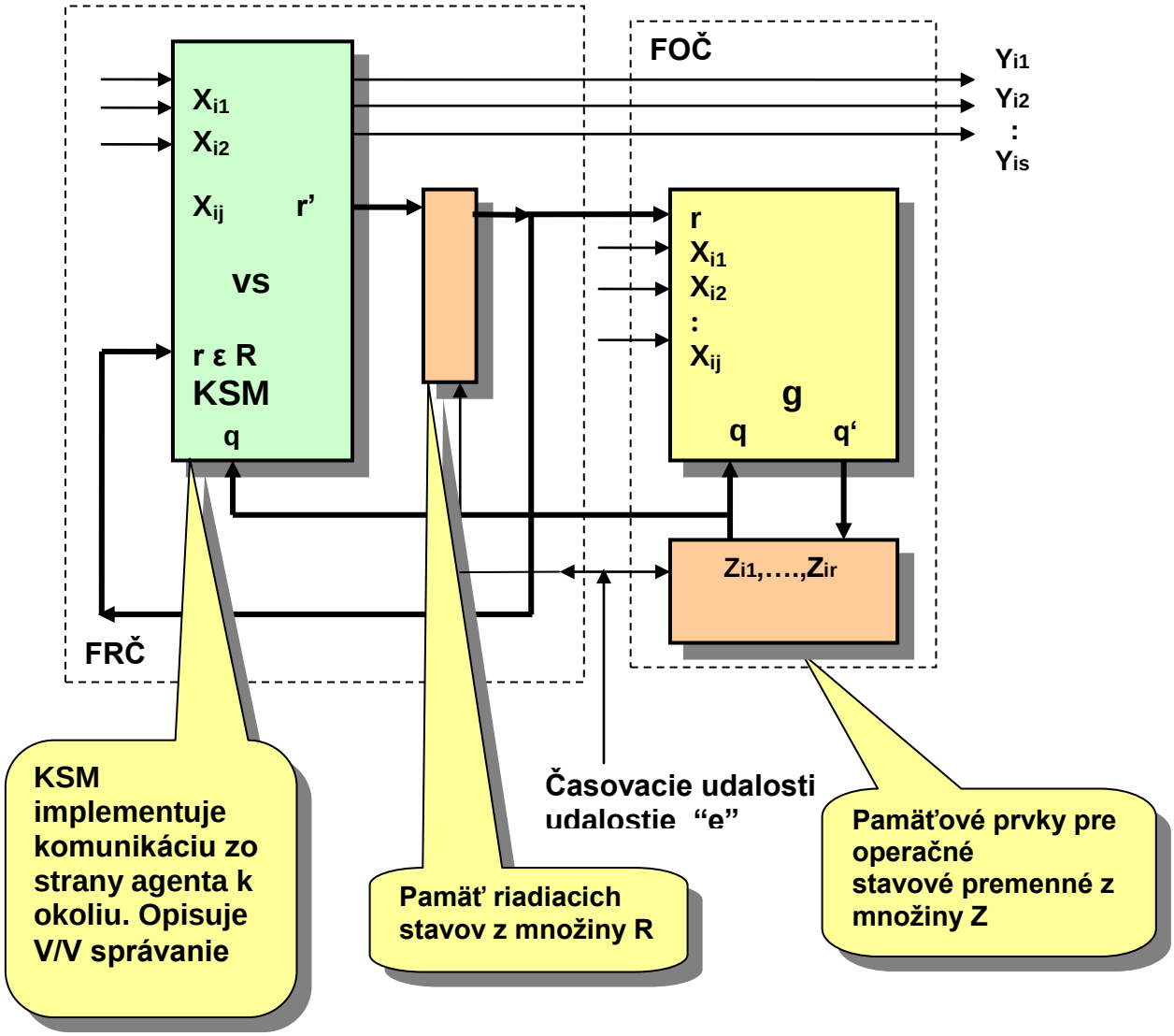
$$\mathbf{g} : \mathbf{SI} \times DV^* \rightarrow \mathbf{Q}$$

funkcia **g** priraďuje (vracia) pre každý začiatočný stav „s“ a každé vstupné slovo v určitý operačný stav agenta q ϵ **Q** Stav q je daný vektorom vektormi hodnôt operačných stavových premenných agenta A, t.j. stavových premenných z množiny operačných stavových premenných Z systému. Agent A v skutočnosti **nastavuje** často hodnoty iba **časti** týchto premenných. Pri danom agente A túto podmnožinu množiny Z všetkých operačných premenných môžme nazvať aj **lokálnou** a potom množinu Z **globálnou množinou** (operačných stavových premenných)**.**

Operačné stavové premenné môžu figurovať zároveň ako **výstupné** primárne premenné (porty) systému, teda pre niektoru výstupnú premennú Yi systému a lokálnu stavovú premennú Zj platí Yi = Zj v každom čase.

Treba poznamenať, že v agente A sa obyčajne nemusia nenastavovať hodnoty všetkých primárnych výstupov (portov). Tie, ktoré A nastavuje môžeme nazvať **lokálne primárne výstupy** (porty) pre A.

Na **architektúru** časti systému, ktorého veličiny sa daného agenta týkajú v celom systéme, možno nazerať ako na štruktúru, ktorá je zobrazená na nasledujucom obrázku. Stavový stroj **KSM** inplementuje riadenie, pri ktorom sa dosahuje špecifikované správanie (prechody medzi riadiacimi stavmi z množiny **R** a generovanie výstupnych slov pre vstupné slová obsiahnuté v množine komunikačných slov, pričom jeho diskrétny čas určujú **časovacie udalosti** v jednotlivých akciách komunikačných slov. KSM realizuje teda sekvenčnú funkciu „**vs**“. Blok **g** reprezentuje pre A relevantnú časť operačnej časti (operačných častí) a stará sa o nastavenie hodnôt lokálnych operačných premenných viditeľných na konci exekúcie agenta A**. FOČ** a **FRČ** sú časti (fragmenty) OČ a RČ systému.



Agent A možno formálne opísať aj jeho kompletným **algebrickým modelom**

A =def (DV,**S**, **SI**, DH, p, v),iniciaálnym konečným Mealyho alebo Moorovým stavovým strojom (so zadaným **SI**), kde **S** je množina vyššie spomenutých globálnych stavov, **SI** je množina začiatočných (globálnych) stavov, "p" a "v" sú jednoduché funkcie: prechová resp. výstupná funkcia: p : **R** x DV → **R** v : **R** x DV → DH. Takýto opis sme uviedli už predtým pri opise agentov Agent-1 a Agent-2 pri CPU.

Pri formálnom opise agentov v **praktických aplikáciách** v ďalšom (namiesto stavového stroja **KSM**) budeme pomocou špecálneho (regulárneho) výrazu (formuly) zadávať iba opis komunikačnej množiny **KM**, ktorá obsahuje všetky komunikačné slová, ktoré sa môžu pri exekúcií agenta objaviť. Množina **KM** kompletne reprezentuje FSM **KSM** a jeho stavový opis (pozri stavový model agenta), ktorej ho možno **algoritmicky odvodiť.** Druhou entitou pri opise agenta bude funkcia **g**.

Pri danom modeli predpokladáme že existuje preň riadiaci vyšie uvedný stavový model

A = $_{def}$ (**SI**, **KSM**, **R**,**vs**),

ktorý mu zodpovedá.

Spôsoby praktického zápisu funkcie g

Fukciu **g**, ktorá vracia vektory hodnôt lokálnych operačných stavových premenných rozpisujeme na **jednotlivé stavové premenné**, ktoré **g** v agente nastavuje (mení) hodnotu. Používame pritom priraďovacie príkazy typu:

$Z_i := \mathbf{g}_i(q,w)$; i=1,..,k

kde $\mathbf{g}_i$ je funkcia, ktorá vstupnej časti komunikačného slova w, teda v1...vn a začiatočnému stavu „s“ priradí hodnou **operačnej premennej** Zr na **konci exekúcie agenta**, teda pri akcií s koncovou časovacou udalosťou ef v každom aktuálnom komunikačnom slove w.

Je zrejmé, že priraďovacie príkazy a funkcie **gi** je potrebné uvádzať iba pre **lokálne** operačné stavové premené, ktorých hodnoty na konci pôsobenia agenta sa v čiastkovom správaní špecifikovanom v agente nastavujú.

Tak **naríklad**, pri systéme VSYS z kap.1 na konci agenta ReadIn sa nastaví iba stavová premené M takto:

M := d1,...,d16,

kde d1,...,d16 sú všetky hodnoty (čísla) vstupu D, ktoré sa objavia vo vstupných častiach VV slov v danej KM. Hodnota stavovej premennej DO sa nezmení.
Komunikačná množina je vo všeobecnosti **nekonečná**. Možno ju však opísať pomocou **konečných formúl**. Formuly, ktoé môžeme použiť sú známe **procesné formuly**. Pri konečných stavových strojoch KSM v agentoch sú tieto formuly (výrazy) **regulárne**. Budeme používať špeciálne procesné formuly, ktoré tu opíšeme. Najskôr však opíšeme určité zjednodušenia pri písaní slov.

Zjednodušenia v zápisoch komunikačných slov

Zavedieme zjednodušenia pri zápise slov.

1. Vektory (v;e;h), nazývame „akcie“. **Akcia** obsahujú vhodné **opis** vstupného vektora „v“ a výstupného vektora „h“ v čase t(e). Budeme ich písať v tvare:

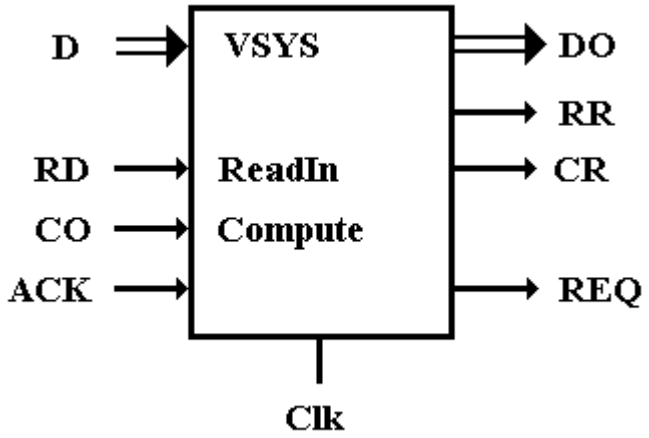
(A=a,...,G=g; e ; I=i,..., Y=y)

kde A,...,G a I,...,Y sú práve **iba** tie primárne vstupy rep. Výstupy systém, ktorých konkrétne hodnoty a,...,g, resp. i,...,y **je potrebné špecifikovať**, pretože ich hodnoty sú pre opis správania v čase t(e) **rozhodujúce**.
Premenné, ktorých hodnoty **netreba špecifikovať** (napr. X, torá vystupuje v akcií takto: X = u) sa teda vynechajú. „Hodnoty“ týchto premených sú nešpecifikované hodnoty "u", ktoré pridávame do domény každého použitého údajového typu.
Ak v niektorej akcií ani jeden vstup (alebo výstup) nie je špecifikovaný, potom namiesto vstupného (alebo výstupného) vektora v (h) v akcií je vektor u (u = (u,..,u)). Uvedené

skrátené zápisy akcií vyjadrujú vlastne **podmnožiny množiny** a to všetkých tých akcií, ktoré v nešpecifikovaných vstupoch a výstupoch nadobúdajú ľubovoľné hodnoty z domény príslušného údajového typu.

Tak napr. v sytéme VSYS možu sa objaviť v niektorom čase takéto **akcie**:
(**RD=1; up(Clk=1); REQ=0), (D=d3,ACK=1; up(Clk=1); REQ=1)**
(u; up(Clk=1); RR=1), (CO=1; up(Clk=1); u) a pod.

Zápis **(RD=1; up(Clk=1); REQ=0)** opisuje vlastne relatívne mohutnú množinu akcií
{ (RD=1,CO= u, ACK=u D=u ; up(Clk=1); REQ=0, RR=u, CR= u, DO=u) },
v ktorej je namiesto „**u**“ dosadená ľubovoľná hodnota CO=0 alebo CO=1,….,DO= d, kde d je ľubovoľný údaj zo 16 celých čísel v danej doméne údajového typu premennej DO, atď.



2. Pri **opise synchrónnych systémov** časovacie udalosti **up(CLK,1)** ako všeobecnú (ako „default“udalosť) v akciách **vynechávame**, t.j., napr. namiesto **(R=1; up(CLK=1); REQ=0)** píšeme zjednodušene: **(R**=1; ;REQ=0).

3. Akcie typu **(...,X=d,..;up(X=d);...)** zapíšeme **jednoduchšie** takto:
(.. ,X=d,..; ; ...),
t.j. **vypustíme** zápis časovacej udalosti a príslušnú premennú **X**, v ktorej udalosť **up(X=d)** je **časovacou udalosťou** v danej akcií, podčiarkneme. Takýto spôsob je zaujímavý pri asynchrónnych systémoch a pri špecifikáciách na vyššej systémovej úrovni, kde metóda definície diskétneho času nie je ešte zaujímavá, t.j. nie je špecifikovaná; a teda určečnie d-bodov času ponecháme iba na zmeny relevantných premenných.

Tvorba Formúl a definícia komunikačných výrazov

Nech ***A*** je **množina akcií** na niektorom digitálnom systéme **Act** = {a,b,c,d,…} akcie a f je **koncová akcia**, ktorá je z množiny **FAct** s takou vlastnosťou, že obsahuje kocovú časovacú udalosť „ef“ . **FAct** je podmnožina množiny **Act**. Ďalej nech „▫“ je symbol operácie **zreťazenia komunikačných výrazov**, ktorá sa aplikuje na dva komunikačné

slová U = L1**.**f‘, V = a**.**L2**.**f , pričom U sa kočí koncovou akciou f’ a V sa začina akciou „a“ a končí koncovou akciou f. Výsledkom zreťazenia U a V je komunikačná formula:

U ▫ V = L1**.**a**.**L2;f

Tu sme jednoduchú konkatenáciu (klasické zreťazenie) , pri ktorom dochádza ku priamemu napojeniu spojeniu v vyjadrili symolom „ **.** “ , teda pre výrazy K a L platí K**.** L = KL (jednoduco besprostredne za sebou pojené výrazy)

Treba si všinúť, že definovanú konkatenáciu „▫“ komunikačných slov budeme v čase interpretovať tak, ako je to naznačené na obrázku.

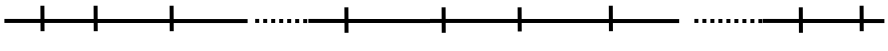
U = L1**.**f ‘

L1 = a**.**b**.**c**.**…. **.**d

V = r**.**L2**.**f

L2 = s**.**t**.**….. k

Vidno, že definované zreťazenie U1 ▫ U2 spojí komunikačné výrazy **prepísaním** koncovej akcie vo výraze U začiatočnou akciou „a“ vo V. V slove U ▫ V (pozri obr. dole) je o jednu akciu menej.



Komunikačné formuly (výrazy) **definujeme** rekurzívne takto:

1. Ak f ϵ **FAct** je komunikačná formula (ďalej len formula)

2. ak a ϵ **Act** - **FAkt** a f ϵ **FAct,** potom **a. f** je formula

3. ak U, V sú formuly, potom aj formuly sú aj

U ▫ V (zreťazenie formúl)

U + V (alternatívne spojenie)

U*, $\mathbf{U^{+}}$ (iterácie, t.j. ľubovoľný konečný poet opakovaní zreťazení „▫“U za sebou)

U^{n} (n násobné opakovanie zreťtazenia „▫“U za sebou)

4. ak U je formula, **X1**, **X2**,…, **Xm** sú premenné systému a <**1**,**n**> je interval indexov 1, 2, …..,n, potom aj $\mathbf{U^{1\ -\ n\ (X1,..,\ Xm)}}$ je formula
(opakovanie ako pri U^{n} , avšak s rôznymi hodnotami premenných **X1**, **X2**,…, **Xm** pri jednotlivých pakovaniach)

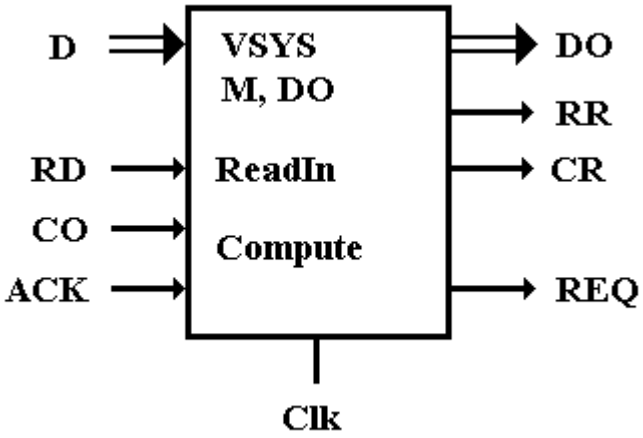
5. ak **U**1 **U**2,…,**U**r sú formuly a **p**1,**p**2,..,**p**r sú **predikáty** definované na premenných systému, ktoré tvoria úplný **disjunktný systém** (t.j. or (**p**1**,** **p**2,...**,** **p**r) = 1 a pre **dva rôzne** predikáty **p**i a **p**j z daného systému predikátov je and (**p**i, **p**j) = 0, potom

p: U # r: V # ……# x: Z

je tiež formula (výber z **U**1 **U**2,…,**U**r podľa hodnoty predikátov **p,r,...,x,** podobne ako pri ríkaze „case of“)

6. ak U a V sú formuly, potom aj **U || V** je fotrmula (paralelná kompozícia formúl U a V)

PRÍKLAD 3 (komunikačného výrazu):



Komunikačnú množinu **KM** agenta ReadIn v systéme VSYS, ktorá je nezávisá na začiatočnom stave z **SI**, možno opísať nasledujúcim komunikačným výrazom:

(RD=1; ;REQ=0).(ACK=1; ;REQ=0)* .[(ACK=0; ;REQ=0)(ACK=0; ;REQ=1)$^{+}$.
(D=d,ACK=1; ;REQ=1).(D=d,ACK=1; ;REQ=0)(ACK=1; ;REQ=0)*]$^{1\text{-}16(D)}$.
(u; ef; RR=1)

(**bodku** medzi akciami, ktorá označuje **jednoduché** zreťazenie akcií, často **vynechávame**) Vo tomto výraze sme vynechali indexovanú postupnosť up(CLK=1, i) ako „default“.

POZNÁMKA: Pri opisoch komumikačných množín pri nekonečných procesoch používame výrayz typu (**U**)$^{\omega}$ (nekonečné opakovanie zreťazení „▫“ komunikačnej formuly U za sebou)

Interpretácia Komunikačných Výrazov

Každý **komunikačný výraz** môžeme zobraziť na množinu komunikačných slov (interpretovať ho ako **určitú množinu** komunikačných slov - v ďalšom píšeme často iba "slov"). Ak uvedieme interpretáciu komunikačných formúl ako zobrazenie

Int : MKF → MKSM,

kde **MKF** je množina komunikačných formúl a **MKSM** je množina všetých možných komunikačných množín KM, potom **Int**(U) pre komunikačnú formulu U je množina KM komunikačných slov, ktoré formula U opisuje (vyjadruje)

- **Int**(U ▫ V) **= Int**(U) ▫ **Int**(V)**,** kde „▫“ značí definované **zreťazenie** komunikačných slov z **Int**(U) a **Int**(V)**,** t.j. **Int**(U ▫ V) = { w_A ▫ w_B | w_A ϶ **Int**(A), w_B ϶ **Int**(B) }
- **Int**(U + V) = **Int**(U) ∪ **Int**(V) („U“ je zjednotenie množín slov)
  Predpoklad sa, že najmenej prvé akcie formúl U a V sú **nekompatibilné** (pozri ďalej)
- FA $\mathbf{p_i}$ ϶ {$\mathbf{p_1}$,.., $\mathbf{p_k}$ }: **Int**($\mathbf{p_1}$: U # $\mathbf{p_2}$: V # ……. # $\mathbf{p_k}$: Z) = **Int**l(A_i) <==> $\mathbf{p_i}$ =1
- **Int**(V*) = **λ** ∪ **Int**(V) ∪ **Int**(V) ▫ **Int**(V) ∪ **Int**(V) ▫ **Int**(V) ▫ **Int**(V) ∪ .....
  kde **„λ“** je tzv. **prázdna akcia**, t.j. ide o matematickú abstrakciu, komunikačné slovo s dĺžkou lg=0. U* je ľubovoľný **konečný počet** opakovaní (**iterácií**) formuly U, vrátane

žiadneho opakovania; predpokladá sa, že vo formule V* ▫ U je prvá(é) akcie **nekompatibilná**(é) s koncovou akciou vo V.

- **Int**(V^+) = **Int**(V) ∪ **Int**(V) ▫ **Int**(V) ∪ **Int**(V) ▫ **Int**(V) ▫ **Int**(V) ∪ ….

  teda zrejme platí: **Int**(V^+) = **Int**(V▫ V*) =; ide o ľubovoľný konečný počet opakovaní a V a vyskytne **aspoň raz**.
- **Int**(V^n) = **Int**(V) ▫**Int**(V) ▫.....▫ **Int**(V) **n - násobné** zreťazenie
- **Int**($V^{1 - n (X1,…,Xm)}$ = **Int**(V^1). **Int**(V^2).....**Int**(V^n), (detto ako pri **Int**(V^n) až nato, že V^j, j=1,2,..,n, značí formulu, ktorá vznikne z V kde sa symbolická hodnota „d“ v každej z premenných X sa v akciách v množine slov z **Int**(V) dosadí **indexovaná hodnota** d_j.

  Symbol 1 - n tu značí **interval** indexov.<1,n>. Teda pri **j – tom opakovaní** niektorého komunikačného slova z komunikačnej množiny opísanej formulou V sa pri zápise hodnoty premennej X z uvedenej množiny (X1,…Xm) vo výraze V sa objaví X = d_j.
- (U || V) = **Int**(A) || **Int**(V) …. (**paralelné** (**súbežne**) vykonávané komunikácie

PRÍKLAD 3 (pokračovanie):
Uvedený komunikačný výraz zo systému **VSYS** pre agent ReadIn

(RD=1; ;REQ=0)(ACK=1; ;REQ=0)*. [(ACK=0; ;REQ=0)(ACK=0;REQ=1) $^+$.

.(ACK=1, D=dj ; ; REQ=1)(ACK=1, D=dj; ;REQ=0).(ACK=1; ;REQ=0)*]$^{1-16(D)}$
.(u; ef; RR=1)

Interpretujeme ho ako množinu komunikačných slov, ktoré tvoria komunikačnú množinu **KM** v agente ReadIn vo VSYS. Treba si všimnúť, že tu sa **16 razy opakuje** komunikačné slovo z výrazu v hranatých zátvorkách a pritom v každom opakovaní sa zvýši index pri symbolickej hodnote „d“ vo vstupnom porte D o jednotku (pozri nižšie).

Po rozvinutí opakovacieho cyklu dostaneme takúto štruktúru výrazu:

(RD=1; ;REQ=0).(ACK=1; ;REQ=0).....(ACK=1; ;REQ=0). [(ACK=0; ;REQ=0).
.(ACK=0; ;REQ=1).(ACK=0;REQ=1)…..(ACK=0; ;REQ=1).(ACK=1, D=d_1 ;REQ=1)

.(ACK=1, D=d_1;;REQ=0).(ACK=1; ;REQ=0)…. (ACK=1; ; REQ=0]

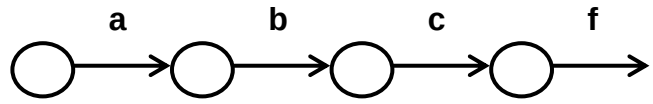
a potom **ešte 15 krát** opakovanie výrazu [..] s inkrementovaním indexu **j** na 2, 3,…,16 hodnoty „**d**“ vstupného portu **D**, pričom záverečný sufix všetkých kom slov bude:

[(ACK=0;;REQ=0)(ACK=0;;REQ=1)(ACK=0;;REQ=1).....(ACK=0;;REQ=1)
(D=d_{16},ACK=1;;REQ=1)(D=d_{16},ACK=1;;REQ=0)….(ACK=1; ; REQ=0] (u;ef;RR=1).

Symbolika **"akcia.....akcia"** tu značí ľubovoľný alebo aj žiadny počet opakovaní akcie, čo vyplýva z **iterácie** „*****“ resp. „$^+$“ . Pri "žiadnom" opakovaní to chápeme ako **vypustenie akcie** na danom mieste zo slova.

Uvedieme ešte ***princíp algoritmickej transformácie*** komunikačných výrazov na konečný stavový stroj (FSM) KSM agenta. Túto transformáciu sme už použili Použijeme pritom jednoduché príklady.

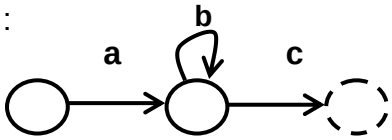
1. Nech **a**, **b**, **c** sú akcie z množiny **Act** - **FAct** a **f** je koncová akcia z **FAac**t. Potom kom výraz **a.b.c.f** sa transformuje na tento Mealyho FSM:



a.b.c.f **b.c.f** **c.f** **f**

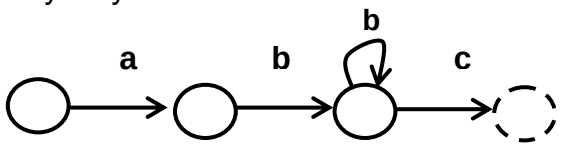
Pri transformácií začneme od stavu ktorý priradíme celému výrazu **a.b.c.f**. Vyberieme **prvú akciu** „**a“** a vytvoríme prechod do nasledujúceho stavu, ktorý zodpovedá sufixu (**b.c.f**) a ktorý nasleduje za **a** vo východiskovom výraze (**a.b.c.f**). Akciu „**a“ pripíšeme** k tomuto prechodu. Rovnakou metódou postupujeme ďalej. Vytvoríme ďalší prechod do stavu, ktorému zodpovedá sufix zvyškového výrazu po jeho prvej akcií **b** a k prechodu pripíšeme akciu **b**, atď. (pozri obrázok hore).

2. Majme výraz **a.b*.c.** (c môže byť koncový akcia).Transformáciou dôjdeme k nasledujúcemu FSM :



a.b*.c **b*.c**

3. Nech **a.b**$^{+}$**.c** je výraz, jeho transformáciou prídeme k nasledujúcemu FSN. Výraz možno nahradiť ekvivalentným výrazom **a.b.b*.c** a tento transformovať.



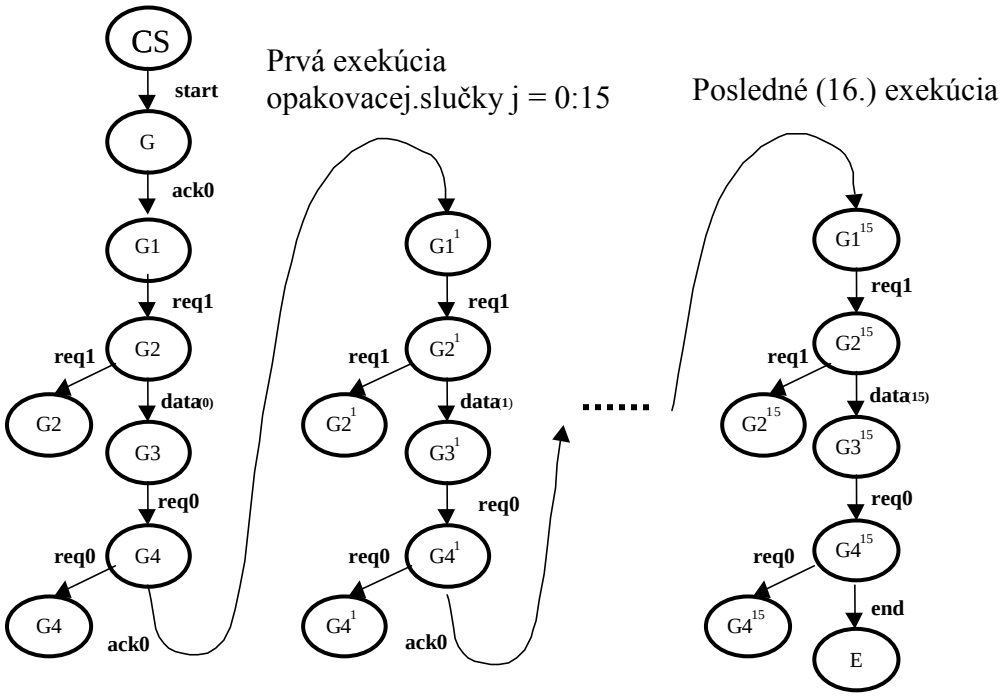
a.b.b*.c **b.b*.c** **b*.c**

4. Teraz ešte zostavme FSM pre nasledujúci kom výraz kom množiny opísaný nižšie. V tomto výraze ktorý opíšeme vo vlastnom formálnom **jazyku HSSL** uvedieme názvy akcií, ktoré jednotlivo opíšeme. „**CS“** tu v HSSL značí komunikačnú množinu **KM** niektorého agenta Readin systému VSYS . „**iv**“ a „**ov**“ sú opisy vstupných resp. výstupných vektorov a „**te**“ je časovacia udalosť v akcií.

CS {

```
{ start.(ack0. req1+.data .req0+)#(j=0:15) . end } //vyraz „KM"
action start { iv: St=1; te: up(Clk,1); }   // opis akcii
action ack0 { iv: Ack=0; te: up(Clk,1); ov: Req=0 }
action req1  { iv: Ack=0; te: up(Clk=1); ov: Req=1 }
action data  { iv: Ack=1, D = d[j]; te: up(Clk,1); ov: Req=1 }
action req0  { iv: Ack=1; te: up(Clk,1); ov: Req=0 }
action  end  { iv: Ack = 0; te:  ef;  req=0}
}
```

Komunikačnému výrazu `{start.(ack0. req1+.data .req0+)#(j=0:15). end }` algoritmus transformácie na FSM vracia nasledujúci **prechodový graf :**



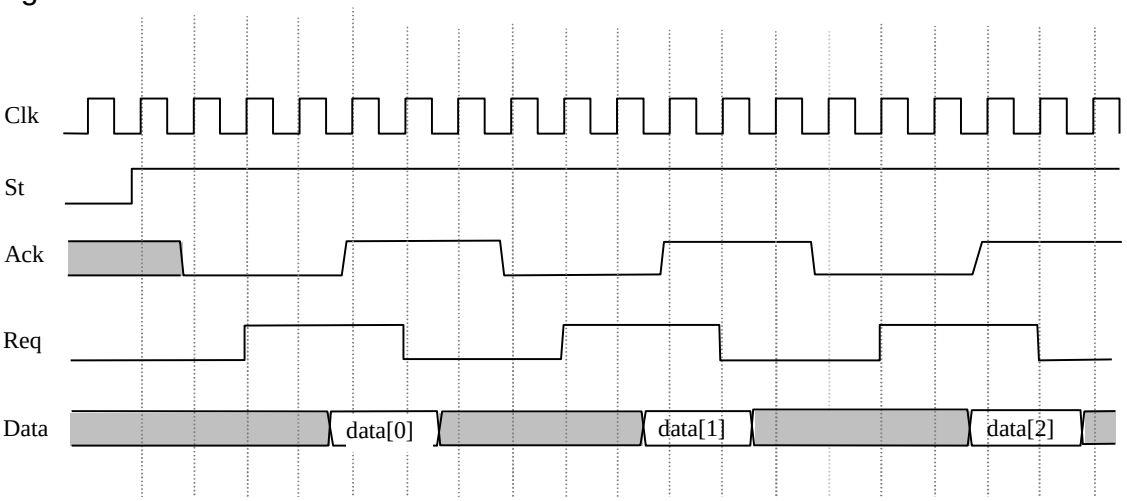
Prechody zo stavu do stavu (napr. pri stave G2) sú vyznačené prechodom z G2 do G2, pričom cieľový stav prechodu je nakreslený ešte raz.

Prechodový graf možno **zjednodušiť** „skrátiť" ak zavedieme **indexovanú vnútornú slučku** (napr. typu „do until **j=0**) začínajúcu s nastavením **indexu j = 0** s inkrementivaním **„j"** pri každom behu o **„+1**" (v module 16) s testom na pravdivosť predikátu (logickej podmienky **j = 0**)na konci slučky. Index **j** možno vsunúť ako vstup (ako prídavok do vstupnej časti) príslušných). Index **j** zodpovedá indexu pri údaji `d[j]` vstupu `D`. Tým odstránime opakovanie vyznačenie slučky `[….] # (j=0 :15)` vo výraze `CS`

Prechodová tabuľka stavového stroja v riadkovej forme, ktorý (ako sa dá dokázať) je **KSM pre** daný agent Readin vo VSYS (Príklad 3) je :

Súč.stav	,	vst. vekor	;	výst. vector	;	nasled.stav
CS	,	St = 1	;	u	;	G
G	,	Ack = 0, j=0	;	Req = 0	;	G1
G1	,	Ack = 0	;	Req = 1	;	G2
G2	,	Ack = 1, D=dj	;	Req = 1	;	G2
G2,	,	Ack = 1, j=j+1	;	Req = 0	;	G3
G3	,	Ack = 0, j ≠ 0	;	Req = 0	;	G1
G3	,	Ack = 0, j = 0	;	Req = 0	;	E
E,	,	u	;	u	;	E

Teraz si ešte všimneme **transformáciu kom výrazov** na **časové priebehy** veličín v diskrétnom a v spojitom čase. V agente ReadIn to vyzerá pri čítaní konkrétneho čísla d_1 agentom takto:



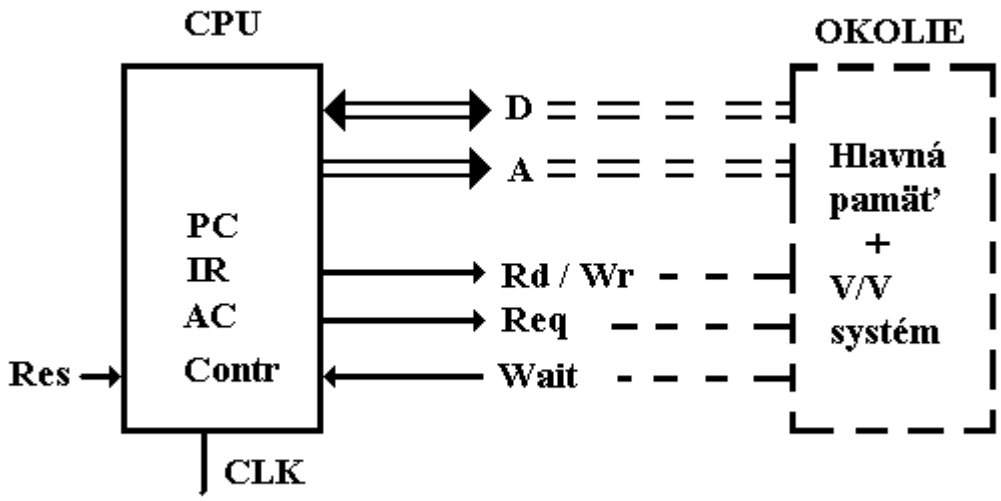
Ak indexujeme d-body po čase udalosti up(St=1) indexami 1, 2, 3,... potom sa opakovací cyklus **opakuje** druhý raz od od d-bodu 8, tretí raz od bodu 14 atď. Celkovo sa opakuje 16 krát (prenesú sa indexovan hodnoty d[0], d[1],..,d[15] zo vstupu D do M).

V prípade komunikácie uvedenej na obrázku ide o takéto **konkrétne komunikačné slovo** (čo do konkrétneho počtu opakovaní v iteráciách "***** "a "$^+$"). Zrejme **patrí** do množiny **KM**

(Start =1, Ack=1;[1];Req=0).(Ack=0;[2];Req=0).(Ack=0;[3];Req=1).
(Ack=0;[4];Req=1).(Ack=0;[5];Req=1)(Ack=1,D=d_1;[6];Req=1)
(Ack=1,D=d_1;[7];Req=0). (Ack=0;[8];Req=0).....atď.

V akciách výrazu sú naznačené indexy d-bodov **1, 2, …**v ktorých sa tieto akcie objavujú. Treba si všimnúť, že z komunikačného výrazu možno odvodiť časový priebeh signálov (hodnôt premenných). Tento priebeh sa dá s **cyklovou presnosťou** získať algoritmickou transformáciou (pri určenom, alebo pri predpokladanom počte iterácií) z komunikačnej formuly. Pri neskoršom doplnení o **časovacie pravidlá** možno generovať časové priebehy aj s presnosťou v **reálnom čase** o čom je potrebné uvažovať na nižších úrovniach návrhu.

PRÍKLAD 4: Procesor CPU, agent Agent-1



V tabuľke opíšeme **detailnú prechodovú tabuľku** stavového stroja **KSM** agenta Agent-1 (vyčítanie inštrukcie pre CPU z externej RAM pamäti) v riadkovej forme.

Stav	Hodnoty vstupov ; hodnoty výstupov	Nasledujúci stav
IF0	Wait = 0 ; u	IF0
IF0	Wait = 1 ; RD/WR=1,Req=1,A=PC	IF1
IF1	Wait = 1 ; RD/WR=1,Req=1,A=PC	IF1
IF1	Wait = 0, D=d ; Req=0	IF2
IF2	Wait = 0 ; u	IF2
IF2	Wait = 1 ; u	OD
OD	u ; u	--

Stavový stroj **KSM** sme odvodili z nižšie uvedenej **komunikačnej formuly** pre Agent-1 formálne opisujúcej komunikáciu tohto agenta s okolím (ktorý vyčíta inštrukciu s uloženou na mieste s danou adresou v registri PC z externej pamäti RAM, a „d.opkod“ tejto inštrukcie prenesie do registra IR). Množinu **R**, prechodovú a výstupnú funkciu „p“ resp. „v“ pre **KSM**, ktoré sú zapísané v tabuľke, je možné **algoritmicky odvodiť** z opisu komunikačnej formuly, ktorá je tu hlavnou špecifikačnou entitou v agente Agent –1 v CPU (**princíp transformačného algoritmu** sme uviedli pri interpretácií kom výrazov vyššie). Komunikačná formula je takáto :

(Wait=0; ;u)*.(Wait=1; ; A=MAR, Req=1,Rd/Wr=1).
.(Wait=1; ; A=MAR, Req=1,Rd/Wr=1)*· (Wait=0, D=d; ; Req=1).
.(Wait=0; ;Req=0)*.(Wait=1, ef, u);

Odvodená množina riadiacich stavov **KSM** je **R** = { **IF0**, **IF1**, **IF2**, **OD**}.
Stav **IF0** je **začiatočný** ,v ktorom je <PC> = a', kde "a'" je (hodnota) adresa pri štarte agenta Agent-1 v PC. V koncovom stave **OD** podľa funkcie **g** platí:

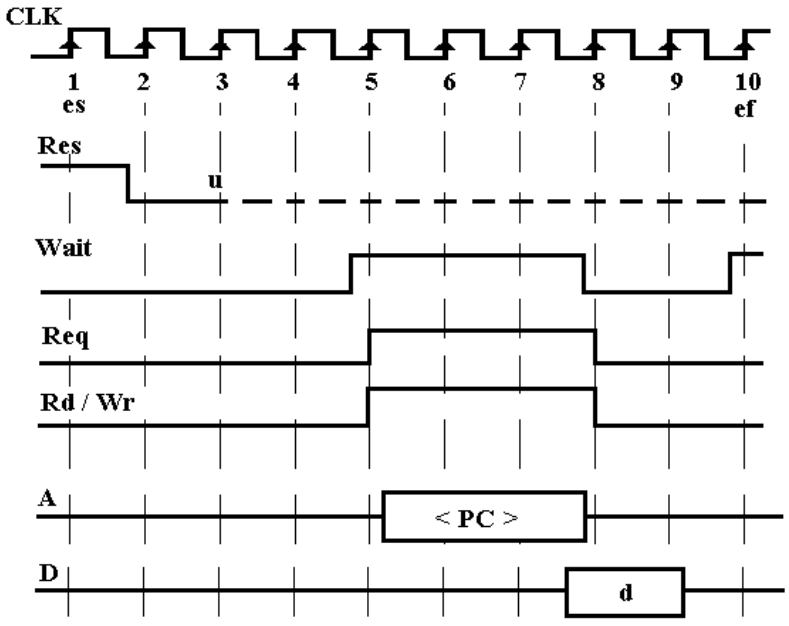
IR := D (<D> = d),
PC := PC + 1,

Výraz **interpretujeme** ako množinu (**KM**) všetkých možných komunikačných slov, ktoré sa môžu indikovať pri vykonávaní agenta Agent-1. „**Vzor**" akéhokoľvek komunikačného slova z tejto množiny je takýto:

(Wait=0; ;u)..... (Wait=0; ;u). (Wait=1; ; A=MAR, Req=1,Rd/Wr=1)…..
.(Wait=1; ; A=MAR, Req=1,Rd/Wr=1).....(Wait=1; ; A=MAR, Req=1,Rd/Wr=1)
.(Wait=0, D=d; ; Req=1).(Wait=0; ;Req=0)…. .(Wait=0; ;Req=0).(Wait=1, ef, u);

Symboly **es** a **ef** tu značia "štartovaciu" (prvú) resp. "koncovú" (poslednú) časovaciu udalosť. Na konci takejto komunikácie, t.j. na konci exekúcie agenta, je systém je v koncovom stave **OD**, ktorý určíme v zhode s funkcia **g** pomocou priraďovacích príkazov pre IR a PC uvedených vyššie.

Časové priebehy premenných zodpovedajúce komunikácií môžu v d-čase a aj v spojitom čase vyzerať takto:



Konkrétne komunikačné slovo zodpovedajúce časovým priebehom, uvedené dole, sa vyvíja od d-bodu 2 (na obr. prvá fáza **Res = 1** v bode 1 vo formule a vprísluš nom FSM nie je uvažovaná)

(Wait=0; $_2$;u) (Wait=0; $_3$;u)(Wait=0; $_4$;u).(Wait=1; $_5$;A=PC, Rd/Wr=1, Req=1).

.(Wait=1; $_6$;A=PC, Rd/Wr=1, Req=1). (Wait=1;$_7$;A=PC,Rd/Wr=1,Req=1).

.(Wait=0,D=d;$_8$;Req=0)(Wait=0;$_9$;Req=0). (Wait=1$_{10}$;ef; u)

Jednoduchý agent, jednocyklový agent - mikrooperácia

V množine agentov sa nachádzajú aj tzv. (komunikačne) jednoduchí agenti.

Jednoduchý agent sa nazýva ten, ktorý má v množine komunikačných slov špecifikovaný práve iba ba **jeden vstupný vektor** a prípadne iba **jeden výstupný vektor**.

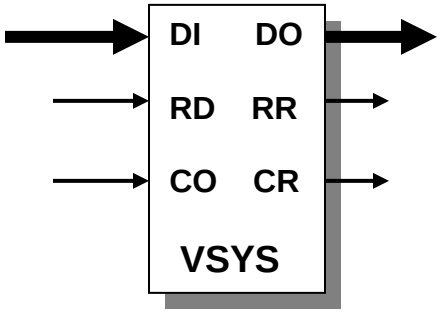
Tak napr. nasledujúci agent ReadIn-0 pri zjednodušenej špecifikácií nášho systému VSYS môže vyzerať takto:

ReadIn-0

KM = (RD=1,DI=d_1,...,d_{16} ; es; u)(u; ;u)*(u ; ef; RR=1)

g: M := d_1,...,d_{16}

Vstupný port **DI** (pozri obr. dole) prenáša celé pole dát d_1,...,d_{16} paralelne na začiatku a ďalšie dáta neakceptuje. Na konci vysiela výstup **RR** = 1



Jednoduchý agent, ktorého VV **slová majú dĺžku 2** sa nazýva **jedno- cyklový.**

Napríklad, nech KM v prípade agenta ReadIn-0 je opísaná takto:

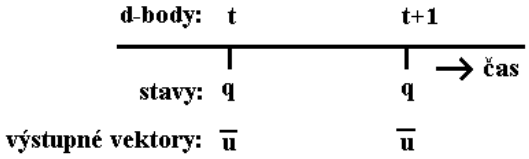
(RD=1, D=d_1,...,d_{16}; ; u)(u ; ; RR=1)

KM má v komunikačnej množine iba dva body diskrétneho času) začiatok a koniec, ktoré sú dané časovacími udalosťami es a ef. Tu sme za časovacie udalosti použili es = up(RD=1) a ef = up(RR=1)).Tento agent ReadIn-0 je jedno – cyklový.

Mikrooperáciou obyčajne nazývame jednoduchý (často jedno - cyklový) agent, pri ktorom 2 susedné body diskrétne diskrétneho času vymedzujú najkratší, ďalej už **nedeliteľný** časový cyklus (mikrocyklus). Pri synchrónnych systémoch je daný susednými hodinovými udalosťami, napr. up(CLK,1, j) a up(CLK,1, j +1).
Príklady uvidíme neskoršie!

Prázdny agent EA

Dôležitým agentom je tzv. **prázdny** agent, EA, je **jednocyklový agent**, ktorý "žiadne spracovanie informácie" nevykoná. Platí preň: **SI**=**S**, pre všetky „s“ z množiny **S** je koncový stav opäť „s“ a výstupný vektor h je u (nešpecifikovaný). Teda uchováva stav, nešpecifikuje výstup a trvá **práve jeden cyklus** na danej úrovni opisu správania systému (pozri obr.).**Vstupné vektory** v oboch d-bodoch t a t+1 **môžu byť pritom akékoľvek**.



Dôležitá poznámka: Upozorňujeme na **podstatný rozdiel** medzi (už spomenutým) **nulovým** agentom **NA** (pozri Algebra agentov) a práve definovaným **prázdnym** agentom **EA**. Nulový agent je matematickou abstrakciou; je definovaný (trvá) iba v jednom d-bode času a má takéto vlastnosti: **IS** = **S**, a pre každý začiatočný stav s ϵ **S** je **KM** = {(u; **ef**; u)}, kde **ef** je začiatočná aj koncová časovacia udalosť agenta

C. Procesy

Spriahnutie agentov vyjadrujeme pomocou **procesov** (pozri Algebra agentov). Procesy na rozdiel od agentov a **konečných** procesov môžu byť aj **nekonečné**. V našom modeli však predpokladáme, že každý nekonečný proves možno opísať procesom typu

$$(\mathbf{A})^{\omega}$$

teda nekonečným počtom zreťazení niektorého agenta **A**. Pripomíname, že **každý agent** v našom modeli je **konečný**, presnejšie jeho komunikačná množina KM j množina komunikačných slov, ktoré sú konečné a zakončené koncovou akciou f z množiny koncových akcií daného agenta **A** (KM možno ju transformovať na FSM). Pre konečný proces existuje agent ktorý ho šecifikuje, hovorímr aj, ktorý mu zodpovdá. Treba pripomenúť že proces (A)* je **konečný**.

Prvkami procesov sú agenty a predikáty definované na množine premenných a agentov systému. Najvšeobecnejším spôsobom formálneho opisu procesov sú tzv. procesné formuly (napr. formuly z Milnerovho CCS - Calculus of Communicating Systems). V tejto časti uvedieme spôsoby zápisu procesov, ktoré budeme používať v týchto prednáškach. Pôjde o "imperatívne" procesy, ktoré vyjadrujú **riadiacu štruktúru** programov s uzavretými cyklami typu "while p do proces P" a "repeat P until p" a výberovou alternatívou (rozvetvením)

Procesné (programové) formuly

Definícia procesných formúl pre **konečné procesy** (rekurzívne):

1. Meno agenta **A** je **triviálna** procesná formula (ďalej len formula)
2. Ak **P1** a **P2** sú dve formuly, potom **P1 . P2** je formula (**bodka** vyjadruje **sekvenčné spriahnutie**, zreťazenie procesov, pri ktorom sa slová komunikačnej množiny procesu

zreťazujú operáciou „▫“)

3. Ak **P1**,...,**Pk** (**k**>1) sú formuly a **p1**,...,**pk** sú **predikáty** definované na systéme ako celku tvoriace **úplný disjunktný** systém predikátov (t.j. and (pi, pj) = 0 pre pi /= pj a zároveň or (p1, p2, ….., pk) = 1), potom

p1:P1,...,pk:Pk

je formula (vyjadruje alternatívne **rozvetvenie** procesov)

4. Ak **P** je formula, potom aj ^{p}**[P]** a tiež **[P]**^{p} sú formuly vyjadrujú dva **cykly** "while **p** do **P**" a "repeat **P** until **p**" zavedené ako v programovacom jazyku pascal. Tu treba upozorniť, že v jazyku C sa druhý typ cyklu **[P]**^{p} s testovaním podmienky na konci definuje rozdielne ako v jazyku pascal a to ako „do **P** while neg(**p**)“.

5. Ak **P** je formula, potom aj **[P] ***, **[P]**^{n} sú formuly [vyjadrujú ľubovoľný, ale **konečný počet opakovaní P** (Kleeneho unárna operácia „hviezdička“), resp. **n**-násobné opakovanie **P**]

6. Ak **P1**,...,**Pk** (**k**>1) sú formuly, potom **P1||.....||Pk** je formula (vyjadruje **paralelné**, **súbežné** spriahnutie procesov)

Nekonečné procesy opisujeme výrazom **P1.[P]** $^{\omega}$, v ktorom **P1** a **P** sú konečne procesy prípadne agenty. Nekonečný proces je užitočným špecifikačným nástrojom.

POZNÁMKA: Ak **P** je proces, možno zaviesť aj formulu [**P**]$^{1\text{-}n\ (X1,...,Xk)}$ ktorú interpretujeme ako **n** násobmé opakovanie s tým že v **P** sa indexujú hodnoty vyznačených premenných **X1**,....,**Xk** indexom **j** (**j** = 1,2,..., n) a pri každom opakovaní procesu **P** sa index **j** inkrementuje. Túto formulu však možno ľahko nahradiť niektorým **štandardným cyklom**.

PRÍKLADY 5:

a. P1 = [(RD=1): ReadIn, (CO=1): Compute, ((RD=0)and(CO=0)):EA]$^{\omega}$

kde **(RD=1)** vyjadruje predikát nadobúdajúci hodnotu **1** práve vtedy ak **RD**=**1**. Podobný zmysel má aj **(CO=1)**. Tretí predikát vykazujú hodnotu **0** ak obidva RD a Co sú nulové. Prípad, v ktorom **RD** = **CO =** 1 okolie systému vylučuje a nie je ani vo formule uvedené.

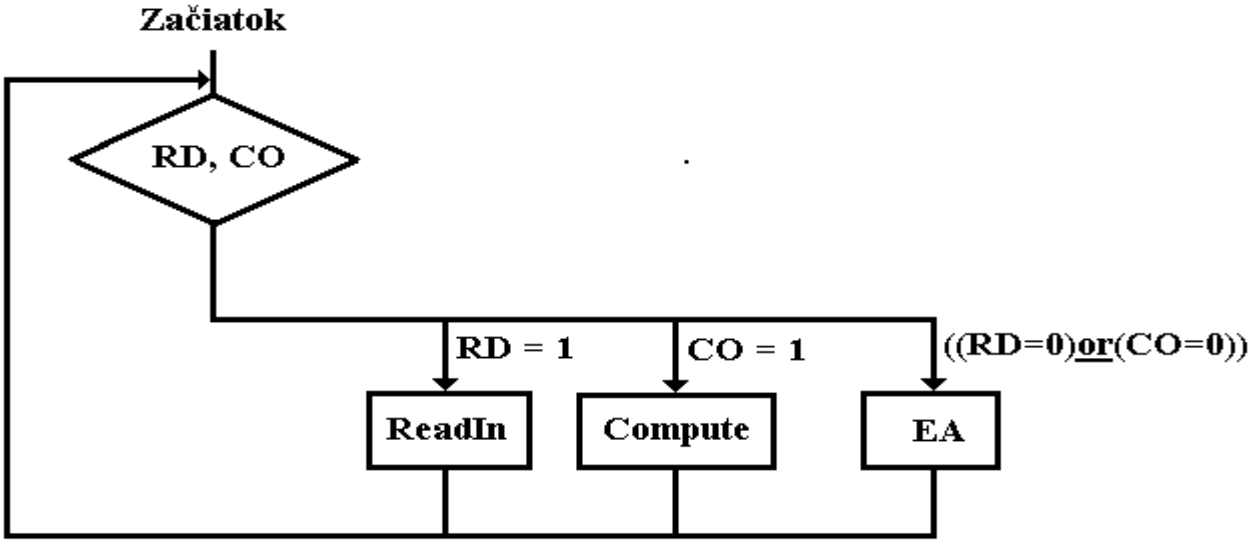
P1 je formula opisujúca proces na systéme VSYS s agentmi ReadIn a Compute. Tu vylučujeme súčasnosť **RD=CO=1**. " $^{\omega}$ " chápeme **nekonečný** počet opakovaní procesu daného formulou medzi zátvorkami [...,].

b. Nech **A1**, **A2**,... sú agenty a **c1**, **c2**,.. sú predikáty definované na niektorom systéme. Potom formula **P2** opisuje proces na tomto systéme (**"-c1"** značí negáciu **c1**).

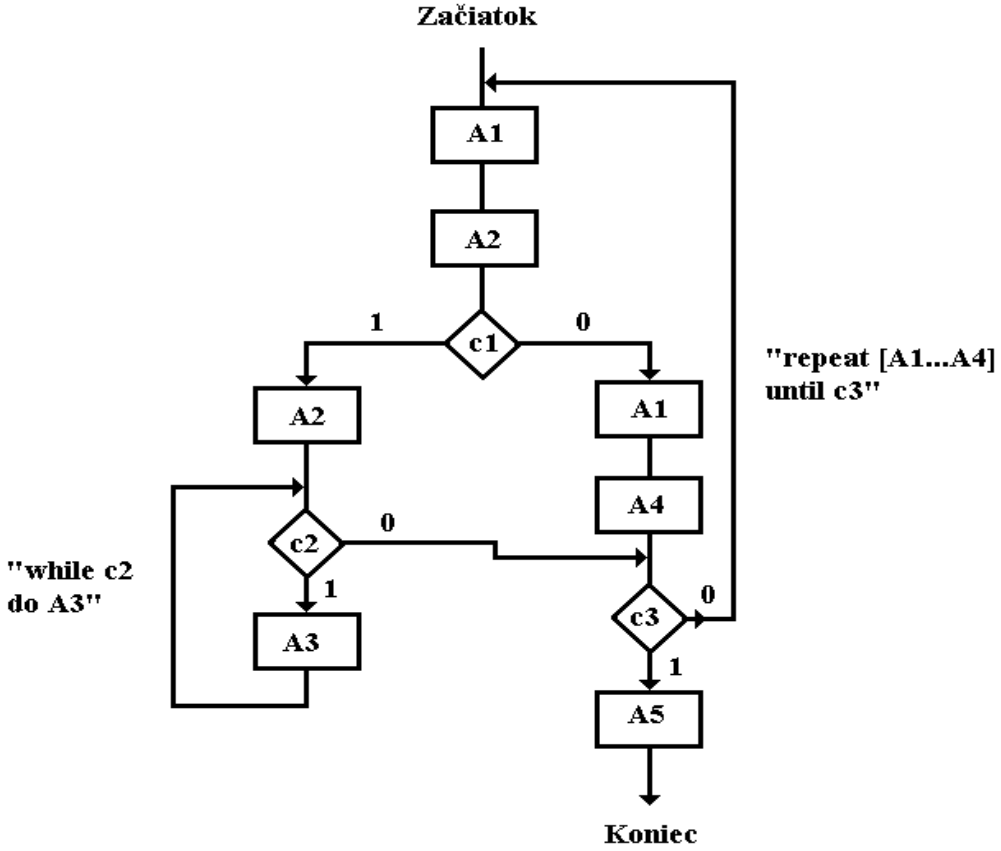
P2 = [A1.A2.(c1:A2.^{c2}**[A3], -c1:A1.A4)]**^{c3}**.A5**

Interpretácia (**sémantika**) formúl je celkovo zrejmá. Ilustrujú ju aj nasledujúce **vývojové diagramy** pre **P1** a **P2**. Upozorňujeme, že v danom prípade sa zreťazujú agenty alebo procesy tak, že ich KM sa zreťazujú operáciou „▫“.

P1 = [(RD=1): ReadIn, (CO=1): Compute, ((RD=0)and(CO=0)):EA]$^{\omega}$



P2 = [A1.A2.(c1: A2.^{c2}[A3], -c1: A1.A4)]^{c3}.A5



Vzťah medzi procesmi a agentmi

O danom **konečnom** procese P môžeme povedať, že **implementuje** niektorý agent A alebo naopak , že agent A je **špecifikáciou** P.

PRÍKLAD 6: Nekonečný proces v systéme **VSYS** môžme vyjadriť ako **P0 = [PrGlob]** $^{\omega}$, kde **PrGlob = (RD=1): ReadIn, (CO=1): Compute, ((RD=0)and(CO=0)):EA** je **konečný** proces, ktorý spriaha agenty **ReadIn**, **Compute** a **EA** a implementuje tak niektorý zložitejší agent ***Glob***, ktorého komunikačnú množinu **KM** môžme vyjadriť takto:

(RD=1): KM_{ReadIn} # (CO=1): $KM_{Compute}$ # (RD=0) and (CO=0): (u;;u).(u;ef;u)

kde KM_{ReadIn} a $KM_{Compute}$ sú komunikačné množiny agentov ReadIn a Compute (ktoré sme už opísali) a komunikačné slovo (u;es;u).(u;ef;u) predstavuje komunikačnú množinu prázdneho agenta **EA**.

Funkciu **g**, určujúcu lokálny operačný stav q na konci agenta ***Glob***, môžeme vyjadriť pomocou priraďovacích príkazov takto:

Ak (**RD=1**), tak **M := d_1,...d_{16}** inak **M := M**

Ak (**CO=1**), tak **DO := M[0] + ... + M[15]** a **M := M**, inak **DO := u**

Agent ***Glob*** môžeme, na druhej strane, chápať ako **špecifikáciu procesu P0**, t.j. opis procesu zvonka, **bez opisu vnútornej štruktúry** procesu (teda konkrétneho spriahnutia jeho prvkov – agentov a opisu jeho prvkov). Agent ***Glob*** teda vyjadruje iba to, **aké celkové správanie** proces PrGlob v systéme VSYS vykazuje, neopisuje vnútornú štruktúru procesu.

Uvedený vzťah medzi procesmi a agentmi využívame pri **tvorivom vývoji**, t, j. pri **zjemňovaní** špecifikácie systému na systémovej úrovni návrhu a tiež smerom zo systémovej úrovne k RT úrovni, ktorá je v našich prednáškach cieľová. Agent **implementujeme** procesom, ktorý spriaha **jednoduchšie** agenty. Na druhej strane možnosť (automatického) zostavenia agenta, ktorý špecifikuje proces na RT úrovni možno účinne využiť pri **formálnej verifikácií** smerom na východiskovú systémovú úroveň.

Tak **napríklad** agent **Glob**-0, ktorý je východiskovou špecifikáciou systému, VSYS implementujeme procesom **PrGlob** = (RD=1): ReadIn, (CO=1): Compute, ((RD=0) and (CO=0)):EA, pričom zavedieme **jednoduchšie** agenty ReadIn a Compute, resp. naopak, ak by sme verifikovali koreknosť zostaveného procesu **PrGlob** voči východiskovému agentovi **Glob**-0, tak porovnáme agent ***PrGlob*** s **Glob**-0.

Programové schémy

Na opis procesov, na **nižšej** RT úrovni špecifikácie, použijeme tzv. programové schémy. **Programovou schémou** nazývame konečnú postupnosť dvojíc (tzv. **označených príkazov** - anglicky: „labeled statements“) **b1:C1; b2:C2; .... ; bn:Cn**, kde **bi** sú symboly nazývané **návestia** (angl. :labels) a **Ci** sú špeciálne formuly, interpretované ako **príkazy** (anglicky: statements).

Príkazy sú **formuly** sú typu **1** - **4** :

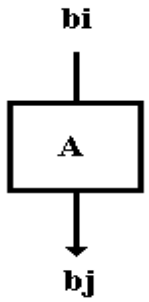
1. **A / b**, kde **A** označuje agent a **b** je návestie z možnej množiny **B** = **{b1,b2,..,bn}** (v zostavovanej schéme)

2. $\mathbf{c_1:A_1/\ bj_1,....,c_k:A_k\ /\ bj_k}$, kde $\mathbf{A_1},..,\mathbf{A_k}$ sú agenty, $\mathbf{c_1},..,\mathbf{c_k}$ sú predikáty tvoriace úplný disjunktívny systém a $\mathbf{bj_1},..,\mathbf{bj_k}$ sú návestia z množiny **B**.

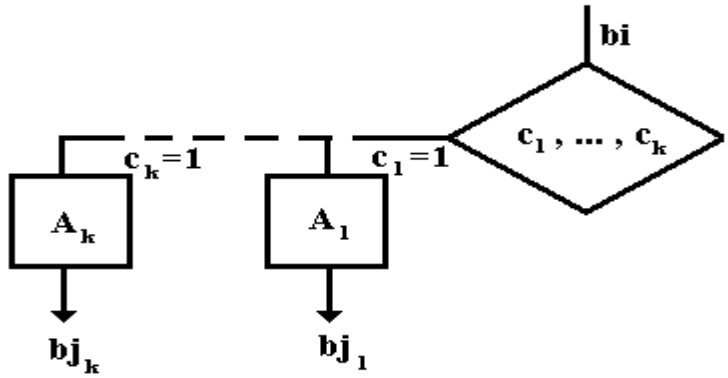
3. $\mathbf{c_1:bj_1,....,c_k bj_k}$, kde $\mathbf{c_1},..,\mathbf{c_k}$ sú predikáty s rovnakou vlastnosťou ako v 2. a $\mathbf{bj_i}$ sú návestia z množiny **B**. V tomto príkaze namiesto každého agenta $\mathbf{A_1},..,\mathbf{A_k}$ si možno predstaviť náhradu **nulovým agentom** NA (Pozor, nie prázdnym agentom EA)
4. **<u>stop</u>** je špeciálny symbol.

Prvý príkaz (príkaz pri návestí **b1**) sa interpretuje ako **prvý**, začiatočný na vykonanie.

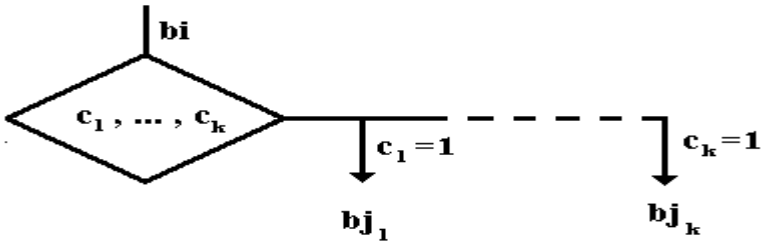
Interpretácia ostatných príkazov je zrejmá z nasledujúcich obrázkov.
1. bi: A / bj (Vykonaj A a potom skoč na návestie bj)



2. **bi:** $\mathbf{c_1/\ A_1:\ bj_1,}$ ....,$\mathbf{c_k:\ A_k\ /\ bj_k}$ (Exekuj agenta **Ai** a rozvetvi podľa hodnôt predikátov)



3. **bi:** $c_1:bj_1,\ldots,c_k: bj_k$ (Iba rozvetvi podľa predikátov)



4. bi: <u>stop</u> (Ukonči proces)

PRÍKLADY 7: Vezmime procesy P1 a P2 z príkladov 5

P1 = [(RD=1): ReadIn, (CO=1): Compute, ((RD=0)<u>and</u>(CO=0)):EA]$^{\omega}$
P2 = [A1.A2.(c1:A2.^{c2}[A3], -c1:A1.A4)]^{c3}.A5

Verzie ich zápisov v programových schémach sú takéto:

P1

N1: (RD): ReadIn / N1, (CO):Compute / N1, (RD) <u>and</u> (CO): EA / N1,

alebo iná verzia zápisu procesu **P1**

N1: (RD): N2, (CO): N3, (RD) <u>and</u> (CO): N4;
N2: ReadIn / N1;
N3: Compute / N1;
N4: EA / N1

P2

L1: A1 / L2;
L2: A2 / L3;
L3: c1: A2 / L4, -c1:A1 / L6;
L4: c2:L5, -c2:L7;
L5: **A3 / L4**
L6: A4 / L7;
L7: c3: A5 / L8, -c3: A1 / L2;
L8: stop

Často je výhodné použiť isté **zjednodušenia** v zápise príkazov.

$\mathbf{c_1: A / bj_1,....,c_k: A / bj_k}$ **==>** $\mathbf{A /c_1:bj_1,....,c_k:bj_k}$

$\mathbf{c_1: A_1 / b,....,c_k: A_k / b}$ **==>** $\mathbf{c_1:A_1,....,c_k:A_k / b}$ operačné rozvetvenie)

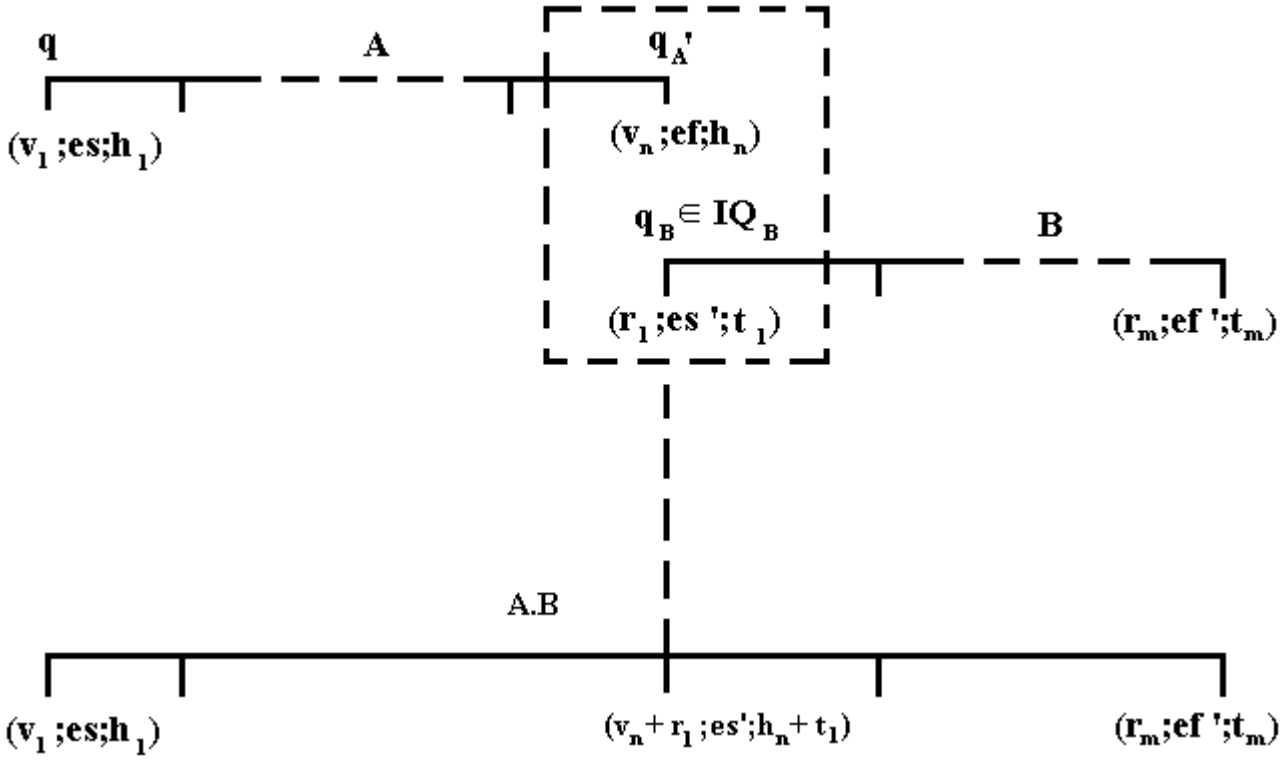
$\mathbf{c: A_1 / bj_1, -c: A_2 / bj_2}$ **==>** $\mathbf{c: A_1 / bj_1, A_2 / bj_2}$ (priame rozvetvenie)

$\mathbf{c: A / bj_1, -c: A / bj_2}$ **==>** $\mathbf{c:A / bj_1, bj_2}$

$\mathbf{c:A_1 / b, -c:A_2 / b ==> c:A_1, -c:A_2 / b}$
Posledné dva príkazy zodpovedajú príkazom **if …then….else**.

Základným typom spriahnutia agentov v procesoch je ich **zreťazenie**. Spôsob zreťazenia agentov obsahuje v sebe ako podstatnú časť **zreťazenie formúl** opisujúcich komunikáciu vo výslednom zreťazení agentov. Zreťazenie komunikačných formúl, ktoré sme už opísali predtým, je založene na prirodzenom princípe, ktorý vyplýva z definície komunikačných množín a je takýto: Majme dva agenty A a B s kom množinami $\mathbf{KM_A}$ a z $\mathbf{KM_A}$, ktoré sa v procese zreťazujú, potom každé kom slovo **w** z $\mathbf{KM_A}$ sa zreťazuje so slovami z $\mathbf{KM_B}$ tak, že sa **zlúči** (stotožní) **koncová** akcia kom slova **w** z $\mathbf{KM_A}$ z **prvou** akciou príslušného slova z $\mathbf{KM_B}$. Teda v súčte bude počte d-bodov v dvoch zreťazených slovách z $\mathbf{KM_A}$ a z $\mathbf{KM_B}$, definovaných časovacími udalosťami v jednotlivých akciách slov, o **jeden d-bod** času **nižší** ako je súčet d-bodov jednotlivých slova z $\mathbf{KM_A}$ a slova z $\mathbf{KM_B}$ pred zreťazením.

Zreťazenie komunikačného slova v $\mathbf{KM_A}$ v agente **A** s komunikačným slovom v $\mathbf{KM_A}$ v agente **B** a výsledné zreťazené slovo v **KM** v **A.B** ilustruje nasledujúci obrázok.



Špeciálna operácia „**+**" tu značí "**zlúčenie**" akcií, ktoré je definované iba ak na každom mieste pre vstupy a výstupy (t.j. pre každú premennú) v akcií sú hodnoty **zlučiteľné**. Pritom **hodnoty** pre jednotlivé vstupy resp. výstupy sú **zlučiteľné** práve vtedy ak sú **rovnaké**, teda ak sú tak **špecifikované** (pripomíname, že domény hodnôt vstupov a výstupov DV a DH obsahujú aj „u"). Potom pre zlúčenie dvoch hodnôt jednotlivých premenných v zlučovaných akciách platí:

a + b

podľa nižšie uvedenej **definície**. Ak hodnot premennej **a**, **b** nie sú zlučiteľné - kompatibilné (pozri nižšie) potom operácia „+" nie je definované

a + a = **a**
a + u = **a**
u + a = **a**
u + u = **u**

Nevyhnutnou podmienkou preto, aby **zreťazenie** agentov **A** a **B** bolo **definované** (a teda malo zmysel) je, aby platilo:
(**a**) Koncová akcia v agente **A** je zlučiteľná so začiatočnou akciou v agente **B**, t.j. vstupné vektory ($\mathbf{v_n}$ a $\mathbf{r_1}$) a tiež výstupné vektory ($\mathbf{h_n}$, $\mathbf{t_1}$) vektory sú zlučiteľné
(**b**) Zreťazenie **A . B** je definované iba pre taký globálny začiatočný stav „$\mathbf{s_A}$" (začiatočný stav agenta **A**) pre ktorý platí:

$$\text{FA } \mathbf{w} \varepsilon \mathbf{KM_A}: \quad \mathbf{q_A} = \mathbf{g_A}(\mathbf{s_A}, \mathbf{w}) \ \varepsilon \ SI_B$$

kde SI_B je množina **začiatočných stavov** v agente **B** a $\mathbf{q_A}$ je koncový stav agenta A pre **w** (teda kocový stav $\mathbf{q_A}$ v **A** padne do množiny SI_B)

Tak **napríklad**, ako vyzerá **zreťazenie formúl** (▫) komunikačných množín agentov ReadIn a Compute v procese **P2 = Readin . Compute** v VSYS.

$\mathbf{KM_{ReadIn}}$ **(Res=1;es;u)(Res=1; ;u)*.(Res=0; ;u).(Wait=0; ; u)*.**
.(Wait=1; ; A=PC, Rd/Wr=1, Req=1)$^+$.(Wait=0, D=d; ;Req=0).(Wait=0;;u)*.
.(Wait=1; ef ; u)

$\mathbf{KM_{Compute}}$ **(u; es; u).(u; ef ; DO=do, CR=1)**

Výsledná kom formula opisujúca **KM** generované v procese **P2** je:

$\mathbf{KM_A} \text{ ▫ } \mathbf{KM_B}$
(Res=1;es;u).(Res=1; ;u)* .(Res=0; ;u)(Wait=0; ;u)*.
.(Wait=1; ;A=PC, Rd/Wr=1, Req=1)$^+$.(Wait=0, D=d; ;Req=0)*(Wait=0;;u).
.(Wait =1; ; u)(u; ef; DO=do, CR=1)

Tu pri zreťazení sme zlúčili koncovú akciu **(Wait=1;ef; u)** v komunikačnej formule agenta ReadIn so začiatočnou akciou **(u; es; u)** v kom formule agenta Compute.

Koncový operačný stav q v konečnom procese **P2**, ktorý vlastne zodpovedá koncovému stavu q agenta **Global** = **ReadIn .Compute** špecifikujúceho **P2**, sa určí funkciou $\mathbf{g}_{ReadIn}(\mathbf{s}_A,\mathbf{w})$ agenta **ReadIn** začiatočným stavom zreťazovaného agenta **Compute**, teda

$$\text{FA } \mathbf{w} \varepsilon \mathbf{KM}_{ReadIn}: \quad \mathbf{q} = \mathbf{g}_{Compute}(\mathbf{s}_B, \mathbf{w}_{Compute}) =$$
$$= (\mathbf{g}_{Compute}(\mathbf{g}_{ReadIn}(\mathbf{s}_A, \mathbf{w}), \mathbf{w}_{Compute}) = \mathbf{g}\,(\mathbf{s}_A, \mathbf{w} \text{ ▫ } \mathbf{w}_{Compute}),$$

kde $\mathbf{s}_A$, je začiatočný stav agenta **ReadIn**, a $\mathbf{s}_B$ je začiatočný stav agenta Compute (a teda koncový stav agenta **ReadIn**) a $\mathbf{w}_{Compute}$ je každé nadväzujúce (zreťaziteľné) komunikačné slovo z $\mathbf{KM}_{Compute}$ na slovo **w** z množiny $\mathbf{KM}_{ReadIn}$.

$$\mathbf{KM}_{Global} = \{ \mathbf{w}_{ReadIn} \text{ ▫ } \mathbf{w}_{Compute} \mid \mathbf{w}_{ReadIn} \in \mathbf{KM}_{ReadIn}, \mathbf{w}_{Compute} \in \mathbf{KM}_{Compute}\}$$

Funkcia **g** agenta Global vracia nasledujúce hodnoty operačných stavových premenných:
M := d1, d2,….,d16 ; DO := d1+ d2+….+d16; CR := 1

Funkcie vs, ktoré určujú hodnoty výstupov v **jednotlivých** kom slovách v Agentoch **ReadIn** a **Compute** sa prenášajú do výsledného zreťazeného agenta **Global** = **ReadIn . Compute**.