PŘIHLÁŠKA VYNÁLEZU

zveřejněná podle § 31 zákona č. 527/1990 Sb.

(21) Číslo dokumentu:
2000 -3910

(13) Druh dokumentu: **A3**

(51) Int. Cl.[7]:
G 06 F 17/30

(19)
ČESKÁ
REPUBLIKA

ÚŘAD
PRŮMYSLOVÉHO
VLASTNICTVÍ

(22) Přihlášeno: **21.04.1999**

(32) Datum podání prioritní přihlášky: **21.04.1998**

(31) Číslo prioritní přihlášky: **1998/19817617**

(33) Země priority: **DE**

(40) Datum zveřejnění přihlášky vynálezu: **11.07.2001**
(Věstník č. 7/2001)

(86) PCT číslo: **PCT/DE99/01208**

(87) PCT číslo zveřejnění: **WO99/55004**

(71) Přihlašovatel:
DETEMOBIL DEUTSCHE TELEKOM MOBILNET GMBH, Bonn, DE;

(72) Původce:
Kehr Klaus, Bonn, DE;

(74) Zástupce:
Všetečka Miloš JUDr., Hálkova 2, Praha 2, 12000;

(54) Název přihlášky vynálezu:

Způsob konverze stromově strukturovaných dat

(57) Anotace:

Popisuje se způsob konverze stromově strukturovaných dat libovolného zdrojového objektového modelu, daných k dispozici prvním zařízením pro zpracování dat, na data volně definovatelného cílového objektového modelu, která se nechají číst druhým zařízením pro zpracování dat, pomocí nezávislého regulačního systému, přičemž zdrojový a cílový strom se skládají z objektů jedné nebo několika tříd objektů, které obsahují vždy jeden nebo několik atributů. Vedle dalších kroků se přitom vytváří především kořen cílového stromu a kořen zdrojového stromu se označuje jako generující instance pro kořen cílového stromu. Dále se stanovují pravidla k vyhledávání výpočetních instancí zdrojového stromu a k výpočtu atributů uzlu zdrojového stromu popř. cílového stromu.

CZ 2000 - 3910 A3

JUDr. Miloš Všetečka
advokát
120 00 Praha 2, Hálkova 2

ZPŮSOB KONVERZE STROMOVĚ STRUKTUROVANÝCH DAT

Oblast techniky

Vynález se týká způsobu konverze stromově strukturovaných dat libovolného zdrojového objektového modelu do volně definovatelného cílového objektového modelu, podle úvodní části patentového nároku 1.

Dosavadní stav techniky

V mnoha oblastech zpracování dat je nutné data, vytvořená a spravovaná nějakým systémem, dát k dispozici jinému systému k dalšímu zpracování. Toto se může uskutečnit výměnou dat v určitém datovém formátu, např. na způsob ASCII-souborů. ASCII-soubory ale často podléhají změnám. Tyto postihují méně „hrubou" syntaktickou výstavbu souboru, než častěji obsahy. Třídy objektů nebo jednotlivé atributy se přidávají nebo chybí, rozsahy hodnot pro atributy se mění, atributy se opět nalézají v jiných třídách objektů. Data se doposud konvertovala do potřebného cílového formátu s pomocí relativně nepružného softwarového řešení. Všechny změny formátu vstupních dat se musely přetáhnout ve zdrojovém kódu a následovně se musela vytvořit nová verze.

Aby se příště mohlo na tyto změny reagovat rychle a pružně, má předložený vynález za úkol, navrhnout způsob řízený pravidly ke konverzi stromově strukturovaných dat, u kterého se mohou zachycovat změny na zdrojových nebo

cílových formátech pomocí jednoduchých změn pravidel.

Tento úkol se řeší charakteristickými znaky patentového nároku 1.

Podstata vynálezu

Vynález je založen na myšlence, „nenalít" pevně pravidla konverze do programového kódu, nýbrž vytvořit regulační mechanizmus, kterým se řídí pružný konvertor v době běhu.

Výhoda, dosažená vynálezem, spočívá při konverzi dat v tom, že se nyní může rychle a pružně reagovat na změny formátu vstupních dat, protože se nepoužívá žádný pevný program, nýbrž přizpůsobitelný regulační mechanizmus.

Vstupní data se přednostně předběžně připravují syntaktickým analyzátorem a v pevném formátu se dávají k dispozici v souboru. Konvertovaná data se rovněž dávají k dispozici v pevném formátu v souboru pro následné zpracování.

Ke zjištění kořenu cílového stromu a kořenu zdrojového stromu se vyznačuje právě ta třída objektu, která leží v hierarchii tříd výše, než všechny ostatní třídy objektů. Každý uzel cílového stromu se vytváří jenom tehdy, když se nechá najít uzel zdrojového stromu, který vyhovuje určitým podmínkám, stanoveným v pravidlu generování. Nalezený uzel zdrojového stromu je potom generující instance pro uzel cílového stromu.

Pro konstrukci nového dílčího stromu pod pozorovaným uzlem cílového stromu se podle vynálezu musí pozorovat jenom dílčí strom pod generující instancí, přiřazenou tomuto uzlu cílového stromu, a cesta od kořenu k této instanci. Pro každou třídu objektu cílového stromu je přitom definováno pravidlo generování.

Na základě pravidla generování se zdrojový strom v okolí generující instance prohledává podle instancí určité třídy objektů, které mají splňovat speciální podmínky, a při úspěšném vyhledání se v cílovém stromu vytváří nový uzel, a instance zdrojového stromu se stává příslušnou generující instancí uzlu cílového stromu.

Pravidla k výpočtu atributů, patřících k nově vytvářeným uzlům (objektům), se přednostně skládají z funkcí, podle kterých se z atributů různých objektů zdrojového stromu počítají atributy nového uzlu cílového stromu.

Aby se umožnila regulovaná konverze, dávají se v regulačním mechanizmu k dispozici pro zdrojový a cílový strom přípustné třídy objektů, přípustní následníci jednotlivých tříd objektů a atributy, příslušející každé třídě objektů (Containment).

Přehled obrázků na výkresech

Vynález bude blíže vysvětlen prostřednictvím konkrétních příkladů provedení znázorněných na výkresech, na kterých představuje

obr. 1 vsazení konvertoru do celkového systému;

obr. 2 struktura konvertoru;

obr. 3 generující instance uzlu K cílového stromu;

obr. 4 vytvoření následníka uzlu cílového stromu;

obr. 5 omezení na dílčí stromy;

obr. 6 vložení nové třídy objektů jako kořenu;

obr. 7 znázornění Containment regulováním;

obr. 8 vstupní body pro prohledávání instancí;

obr. 9 stanovení oblastí prohledávání;

obr. 10 vytváření jenom jednoho nebo několika následníků;

obr. 11 pravidla generování instancí v regulačním mechanizmu;

obr. 12 pravidla výpočtu atributů v regulačním mechanizmu;

obr. 13 vytvářet kořen cílového stromu;

obr. 14 vypočítat atributy nové instance;

obr. 15 vytvářet následníka nové instance;

obr. 16 označení nové instance jako „stará";

Příklady provedení vynálezu

V tomto konkrétním případu se vychází z konverze dat, jak se uskutečňuje v moderních digitálních radiotelefonních sítích, aby např. data, která jsou k dispozici v Operation a Management Center (OMC), dala k dispozici Telekommunikation Management Network (TMN) - platformě. Ve smyslu zavedení TMN-platformy 2 u provozovatelů mobilních radiotelefonních sítí existuje úkol, dát k dispozici konfigurační data základní stanice pro platformově interní MIB (Management Information Base) 3. Na základě nedostatečných alternativ se k tomu přistupuje na ASCII-soubory, které se nechají získat u všech systémových technik základních stanic nástroji, specifickými pro výrobce, z OMC - počítačů 4. Obrázek 1 ukazuje vložení konvertoru 1 do takového systému.

Obrázek 2 ukazuje konstrukci konvertoru 1. Aby mohl konvertor vykonávat svou práci, vstupní data 5 (OMP-data) se předběžně zpracovávají a jsou k dispozici v pevně daném formátu v souboru 6 (předřazený soubor). Tuto úlohu přebírá v tomto případě algoritmus pro syntaktickou analýzu (syntaktický analyzátor 7). Tento předřazený soubor se pro jiné požadavky může lehce nahradit jiným modulem. Konvertor 1 opět vytváří v pevně daném formátu soubor 8, který je potom k dispozici pro následné zpracování (Backend). Zde se mimoto zjišťují rozdíly od dat, generovaných předchozí den (Delta - generování 9) a odchylky se posílají jako CMIS-zprávy (Common Management Information Services-Format) na TMN-platformu 2, kde se zpracovávají do MIB-datového souboru. Také Backend se může vyměnit za jiné účely.

Princip funkce

Způsob funkce konvertoru 1 se odvozuje z následujících podstatných pozorování:

Podle obrázku 3 spočívá rozhodnutí, zda se v cílovém stromě 10 musí vytvářet instance (uzly) 11, na vyhodnocení relativně omezené oblasti zdrojového stromu 12. Přitom se nechá vždy určit instance 13 zdrojového stromu 12, dále označovaná jako „generující instance" EI, jejíž existence se pozoruje jako základní předpoklad k vytváření nové instance 11, uzlu K.

Jak je znázorněno na obrázku 4, nechají se stanovit všeobecně platná pravidla, která se mohou vyhodnocovat v okolí generující instance EI 13 uzlu a nezávisle na konkrétní poloze ve zdrojovém stromu 12, aby se mohl vytvářet následník/následníci A 14 uzlu K 11. V příkladu to znamená: vytvoř uzel A 14, když ve zdrojovém stromu 12 uzel B 15 a C 16 jsou následníci EI 13.

Pro konstruování nového dílčího stromu 17 pod uzlem 11 v cílovém stromu 10 stačí prohlížení dílčího stromu 18 generující instance 13 tohoto uzlu 11 v zdrojovém stromu 12. Toto je objasněno na obrázku 5. Na tomto místě je třeba zmínit, že pro případně se vyskytující zvláštní případy se musí dávat k dispozici funkce, které mohou tyto principy porušovat a umožňují navigování a vyhodnocování podmínek v celkovém zdrojovém stromu 12.

Regulační množiny

Jak již bylo zmíněno, nemá konvertor 1 obsahovat „tvrdě kódovanou" inteligenci k vytvoření cílového stromu 10 ze

zdrojového stromu 12, nýbrž se v době běhu zjišťují a používají předpisy pro konverzi, vhodné pro příslušnou situaci. Tento regulační mechanizmus se v uvedeném příkladu ukládá a interpretuje v ASCII-souboru. Ostatní varianty, jako např. ukládání pravidel v databázi, jsou zde myslitelné.

Pro konstrukci cílového stromu 10 z daného zdrojového stromu 12 potřebuje konvertor pravidla, ze kterých může získat následující informace:

1. Které následníky může mít instance cílového stromu
2. Jaké podmínky musí být splněny, aby se v cílovém stromu vytvářela nová instance jako následník existující
3. Jak vypočítám atributy nové instance cílového stromu

Dodatečně se jeví smysluplné, přezkoušet na správnost pravidla generování instancí a pravidla výpočtu atributů. K tomu jsou zapotřebí následující informace, které se rovněž mohou lehce uložit v regulačním mechanizmu:

1. Jaké následníky může mít instance zdrojového stromu
2. Jaké atributy obsahuje každá instance zdrojového a cílového stromu
3. Jaké rozsahy hodnot platí pro hodnoty atributů

V dalším se blíže popisuje konstrukce regulačního mechanizmu a vysvětluje se, které informace se získávají z příslušných typů regulace. Tyto informace se potom používají popsaným algoritmem způsobu ke konverzi.

Základně důležité pro provedení způsobu a pro to, že se pravidla regulačního mechanizmu vůbec mohou používat, je

konfigurace datových struktur, se kterými se vstupní data reprezentují na způsob zdrojového stromu a výstupní data na způsob cílového stromu. Proto se v dalším nejdříve přistupuje na požadavky na datové struktury.

Požadavky na datové struktury

Jak již bylo zmíněno, zdrojová a cílová data mají existovat na způsob takzvaných „stromů". K tomu se mohou používat datové struktury, v EDV-technice obecně známé. Aby se na těchto datových strukturách mohla používat pravidla regulačního mechanizmu a mohl se provádět základní algoritmus ke konstrukci cílového stromu, je třeba datové struktury koncipovat tak, že se určité informace mohou zjišťovat v každém okamžiku (a pokud možno účinně).

Tyto informace jsou:

- předchůdce každého uzlu stromu (pokud existuje)
- následnici každého uzlu (pokud existují)
- třída objektu, ke které patří každý uzel
- konkrétní instance, která reprezentuje každý uzel
- konkrétní hodnoty všech atributů každého uzlu
- cesta od kořenu ke každému uzlu
- dílčí strom pod každým uzlem
- příslušnost pozorovaného uzlu ke zdrojovému nebo k cílovému stromu
- „generující instance" pro každý uzel cílového stromu

Na konkrétních zdrojových a cílových stromech se nyní s pomocí příslušných vyhledávacích algoritmů mohou zjistit množiny hodnot, na kterých se vypočítává už řada základních funkcí konvertoru:

Množiny hodnot jsou:

K_Q	množina všech uzlů zdrojového stromu (vstup)
K_Z	množina všech uzlů cílového stromu (generující výstup)
$K = K_Q \cup K_Z$	množina všech uzlů stromů

Pro množinu M má Pot(M) označovat příslušnou množinu mocnin.

Poté se mohou vypočítat následující základní funkce:

Každý uzel stromu patří určité třídě objektů. Toto zjišťuje funkce Obj

Obj: $K \rightarrow KL$

Pro každý uzel stromu zjišťuje Sub pod ním ležící dílčí strom

Sub: $K \rightarrow Pot(K)$

Pro každý uzel stromu zjišťuje Path cestu od kořenu k tomuto uzlu

Path: $K \rightarrow Pot(K)$

Pro každý uzel cílového stromu zjišťuje Ei generující instanci ve zdrojovém stromu

Ei: $K_Z \rightarrow K_Q$

Horní základní funkce se realizují jednoduchým prohledáváním stromů. K tomu se musí splnit formulované požadavky na datové struktury.

Zavedení třídy „root" (kořen)

Aby se stanovil definovaný počátek pro vyhodnocení pravidel během konverze a vlastní algoritmus konverze, je nutné třídy objektů označit jako „kořen" zdrojového a cílového stromu. Všechny ostatní třídy objektů musí být dosažitelné po cestě od kořenu a algoritmus začíná s vyhodnocením pravidel pro třídu kořenu cílového stromu. Jak je znázorněno na obrázku 6, realizuje se požadovaná struktura zde pomocí zavedení úplně nových tříd objektů, totiž R_Z (R je pro Root) jako kořenu cílového stromu a R_Q jako kořenu 19 zdrojového stromu. Třídy objektů, nejvyšší v hierarchii tříd, např. A a B, zdrojového popř. cílového stromu, se potom v regulačním mechanizmu stanovují jako následníci R_Q popř. R_Z.

Tento způsob jednání má k tomu vedlejší efekt, že několik, nejprve nezávislých zdrojových stromů se může zároveň konvertovat, zatím co se připojením na R_Q 19 spojují do nového zdrojového stromu 12.

Popis struktury zdrojového a cílového stromu

S pomocí pravidel této kategorie je možno popsat Containment-Tree 20 (obrázek 7) pro zdrojová a cílová data. Musí se nechat odečíst:

1. které třídy objektů se mohou/musí vyskytovat?
Tento požadavek se může realizovat pomocí seznamu, který obsahuje přípustné třídy 21 objektů, např. A, B, ... G. Třídy objektů, které se musí vyskytovat bezpodmínečně, (jako např. nová třída „root" nebo původní kořen stromu), se označují jako „naléhavě potřebné".

2. které následníky, např. B, C může/musí mít každá třída objektů?
Ke každé třídě 21 objektů existuje seznam, ve kterém se

nechají odečíst její přípustní následníci. Také zde se mohou třídy objektů, jejichž nepřítomnost by znamenala výraznou chybu, označovat jako „naléhavě potřebné".

3. které atributy 22 může/musí obsahovat každá třída objektů?
Ke každé třídě 21 objektů existuje další seznam, ve kterém jsou zaznamenány příslušné atributy 22. Také zde je navržena možnost, označovat důležité atributy jako „naléhavě potřebné".

Z těchto pravidel (obrázek 7) se mohou pomocí příslušných vyhledávacích algoritmů odvodit množiny hodnot, ze kterých se již vypočítává řada základních funkcí konvertoru 1:

Množiny hodnot jsou:

KL_Q	množina všech tříd 21 objektů zdrojového stromu 12
KL_Z	množina všech tříd 21 objektů cílového stromu 10
$KL = KL_Q \cup KL_Z$	množina všech tříd 21 objektů regulačního mechanizmu
AT_Q	množina všech atributů 22 objektů 20 zdrojového stromu 12
AT_Z	množina všech atributů 22 objektů 21 cílového stromu 10
$AT = AT_Q \cup AT_Z$	množina všech atributů 22 objektů 21 regulačního mechanizmu

Dodatečně se zde může pomocí údaje rozsahů hodnot pro atributy 22 stanovit také následující množinu:

VAL množina všech možných hodnot atributů

Poté se mohou vypočítat následující základní funkce:

Pro každou třídu 21 objektů zjišťuje funkce Nf množinu všech možných tříd následníků
Nf: KL $\rightarrow$ Pot(KL_Q) $\cup$ Pot(KL_Z)

Pro množinu tříd 21 objektů zjišťuje Att množinu všech atributů 22, obsažených uvnitř

Att: Pot(KL) $\rightarrow$ Pot(AT)

Výše uvedené základní funkce se mohou realizovat jednoduchým prohledáváním množin pravidel.

Pravidla generování instancí cílového stromu

Rozhodující úloha při konstrukci cílového stromu 10 spočívá v rozpoznání, kdy je možno vytvořit instanci 11 třídy objektů cílového stromu. K řešení tohoto problému slouží takzvaná pravidla generování, z čehož se musí udat přesně jedno pro každou třídu 21 objektů cílového stromu 10. Pro každou potenciální třídu následníků uzlu 11 cílového stromu se potom vyhodnocuje příslušné pravidlo generování v „okolí" generující instance 13 uzlu ve zdrojovém stromu 12 a tak se popř. vytvářejí nové cílové uzly 11 jako následníci původního. Pravidlo generování samo „prohledává" zdrojový strom 12 v okolí generující instance 13 podle instance určité třídy 21 objektů, které mají splňovat speciální podmínky. Pokud je hledání úspěšné, vytváří se nový uzel 11 v cílovém stromu 10 a instance cílového stromu k příslušné generující instanci 13 cílového uzlu 11.

V pravidle generování jsou nutné různé údaje:

1. vstupní bod 23 pro vyhledávání (obrázek 8)
Jako vstupní bod se musí stanovit jeden z následujících uzlů zdrojového stromu:
- kořen zdrojového stromu
- generující instance 13
- jiný uzel na cestě od kořenu ke generující instanci

2. Typ vyhledávání (vyhledávací oblast, obrázek 9)
Musí se stanovit jedna z následujících vyhledávacích oblastí:
- prohledává se celý dílčí strom 24
- prohledává se určitá úroveň 25 pod vstupním bodem

3. Dodatečné podmínky
- zde se mohou udávat další podmínky, které má splňovat uzel zdrojového stromu, jako např. se vyznačovat určitými hodnotami atributů, nebo existence dodatečných instancí cílového stromu

4. Četnost použití (obrázek 10)
Pro četnost použití se musí zvolit jedna z následujících alternativ:
- vytvoř uzel cílového stromu pro každý aktivní záznam
- vytvoř jenom jeden uzel cílového stromu u prvního aktivního záznamu

Uzel 11 cílového uzlu se vytváří, jenom když se pro něj s pomocí příslušného pravidla generování nechá ve zdrojovém stromu najít generující instance 13.

Obrázek 11 má ještě jednou znázorňovat zápis v regulačním mechanizmu, potřebný pro zanesení pravidel generování instancí.

Se zavedením těchto pravidel obdržíme novou množinu hodnot a základní funkci:

Množina hodnot je:

ER množina všech pravidel generování v regulačním mechanizmu

Potom se může vypočítat následující základní funkce:

Pro každou třídu cílového stromu zjišťuje funkce Gen příslušné pravidlo generování

Gen: $KL_Z \rightarrow ER$

Také tato základní funkce se nechá realizovat „jednoduchým prohledáváním" množin regulování. Přesný popis se zde proto neuskutečňuje.

Pravidlům generování se tak může rozumět jako funkcím, které pro určitou objektovou třídu zkoumají množinu uzlů zdrojového stromu a popř. zpět poskytují jednu z nich jako generující instanci pro novou instanci pozorované třídy objektů cílového stromu (tzn. nový uzel cílového stromu). Pokud se nenachází žádná generující instance, nevytváří se také žádný uzel cílového stromu.

$er \in ER$

$er: KL_Z \times Pot(K_Q) \rightarrow K_Q$

Pravidla výpočtu pro atributy

Poté co se vytvořil objekt cílového stromu, musejí se nakonec ještě vypočítat atributy, v něm obsažené. K tomu se pro každý atribut udává vzorec. Tento vzorec obsahuje jako operátory běžné výpočetní operace a jako operandy konstanty, proměnné a hodnoty atributů zdrojového stromu. Přitom se může k atributům generující instance (která je k tomuto časovému okamžiku již pevně určena) jednoduše přistupovat podle jména. Atributy tohoto objektu ale vždy nestačí k výpočtu nové hodnoty. Je tedy třeba vytvořit možnost, mít přístup k atributům libovolných dalších objektů. K tomu se ke každé třídě objektů cílového stromu udává, které další instance cílového stromu mají „dát k dispozici" své atributy k výpočtu vlastních hodnot atributů. Tyto instance zdrojového stromu se dále nazývají „výpočetní instance". Výpočetní instance se nalézají podobným způsobem, jako generující instance.

Vyhledávací pravidlo pro výpočetní instanci obsahuje následující údaje:

1. Vstupní bod pro vyhledávání

Jako vstupní bod se musí stanovit jeden z následujících uzlů zdrojového kmene:

- kořen zdrojového stromu
- generující instance
- další uzel na cestě od kořenu ke generující instanci

2. Typ vyhledávání

Musí se stanovit jedna z následujících oblastí vyhledávání

- celý dílčí strom
- určitá úroveň pod vstupním bodem

3. Dodatečné podmínky
   - zde se mohou udávat další podmínky, které má splnit uzel zdrojového stromu, jako např. mít určité hodnoty atributů nebo existence dodatečných instancí cílového stromu

(4. parametr z vyhledávání generující instance, četnost použití, zde odpadá. Vždy se volí první shoda).

Při stanovení pravidel vyhledávání k vyhledávání instancí výpočtu pro třídu cílového stromu obdrží každé pravidlo vyhledávání jméno, jednoznačné v třídě cílového stromu. Pokud by měly různé instance výpočtu obsahovat atributy stejného jména, může se potom atribut jednoznačně zvolit přes konstrukt <jméno instance výpočtu>.<jméno atributu>.

Obrázek 12 má ještě jednou znázornit zápisy v regulačním mechanizmu, potřebné pro zápis pravidel výpočtu atributů. Se zavedením těchto pravidel dostaneme novou množinu hodnot a další základní funkci:

Množina hodnot je:

AR množina všech pravidel pro výpočet atributů v regulačním mechanizmu

Potom se může vypočítat následující základní funkce:

Pro každý atribut uvnitř třídy objektu stanovuje Calc příslušný předpis výpočtu

Calc: $AT_Z \times KL_Z \rightarrow AR$

Jako všechny dosavadní základní funkce se také tato nechá realizovat „jednoduchým prohledáváním" množin pravidel. Přesný popis se zde proto neuskutečňuje.

Pravidlům výpočtu atributů se může tedy rozumět jako funkcím, které z atributů různých objektů zdrojového kmenu vypočítají novou hodnotu.

$ar \in AR$

$ar: Pot(K_Z \times AT_Z) \rightarrow VAL$

Základní algoritmus

Nejdříve se ještě jednou provádějí základní funkce, které se potom používají u popisu základního postupu ke konstrukci cílového stromu. Způsob funkce těchto základních funkcí se už zde neuskutečňuje, protože jejich způsob práce byl vysvětlován již dříve.

Funkce pro regulační mechanizmus

Z daného, správně vybudovaného regulačního mechanizmu (kontroluje se přes Check-Routine) se mohou pomocí jednoduché funkce zjišťovat následující výsledky:

budiž

M	množina
Pot(M)	příslušná množina mocnin
ER	množina všech pravidel generování v regulačním mechanizmu
AR	množina všech pravidel k výpočtu atributů v regulačním mechanizmu
KL_Q	množina všech tříd objektů zdrojového

	stromu
KL_Z	množina všech tříd objektů cílového stromu
$KL = KL_Q \cup KL_Z$	množina všech tříd objektů regulačního mechanizmu
AT_Q	množina všech atributů objektů zdrojového stromu
AT_Z	množina všech atributů objektů cílového stromu
$AT = AT_Q \cup AT_Z$	množina všech atributů objektů regulačního mechanizmu
VAL	množina všech možných hodnot atributů

Pro každou třídu objektů zjišťuje funkce Nf množinu všech možných tříd následníků

Nf: $KL \rightarrow Pot(KL_Q) \cup Pot(KL_Z)$

Pro každou třídu cílového stromu zjišťuje funkce Gen příslušné pravidlo generování

Gen: $KL_Z \rightarrow ER$

Pro množinu tříd objektů zjišťuje Att množinu všech v něm obsažených atributů

Att: $Pot(KL) \rightarrow Pot(AT)$

Pro každý atribut uvnitř třídy objektů zjišťuje Calc příslušný předpis výpočtu

Calc: $AT_Z \times KL_Z \rightarrow AR$

Funkce pro zdrojový a cílový strom

Ve správně konstruovaném stromu (zdrojový nebo cílový strom) se mohou lehce zjistit následující výsledky:

budiž

K_Q	množina všech uzlů zdrojového stromu (vstup
K_Z	množina všech uzlů cílového stromu (výstup, který je třeba generovat
$K = K_Q \cup K_Z$	množina všech uzlů stromu

Každý uzel stromu patří určité třídě objektů. Toto zjišťuje funkce Obj

Obj: $K \rightarrow KL$

Pro každý uzel stromu zjišťuje Sub pod ním ležící dílčí strom

Sub: $K \rightarrow Pot(K)$

Pro každý uzel stromu zjišťuje Path cestu od kořenu k tomuto uzlu

Path: $K \rightarrow Pot(K)$

Pravidlo generování hledá v množině uzlů zdrojového stromu ten, který splňuje všechny podmínky a tím se má stát „generující instancí" pro novou instanci třídy objektů cílového stromu. (Pokud se nenajde žádný vhodný uzel, nevytváří se žádná instance).

$er \in ER$; $er: KL_Z \times Pot(K_Q) \rightarrow K_Q$

Pro každý uzel cílového stromu zjišťuje Ei generující instanci ve zdrojovém stromu

Ei: $K_Z \rightarrow K_Q$

Předpis výpočtu pro atributy zjišťuje z konkrétních hodnot atributů v uzlech stromu novou hodnotu

ar ∈ AR; ar: Pot(K x AT) → VAL

Konstrukce cílového stromu

S pomocí výše uvedených funkcí se může nyní uvést algoritmus, který konstruuje cílový strom 10 s „root" začínajícím „úrovňovým způsobem" shora směrem dolů.

Budiž

R_Z kořen 26 cílového stromu 10

R_Q kořen 19 zdrojového stromu 12

1. vytvoř kořen R_Z 26 jako první „novou instanci" cílového stromu 10 a přiděl kořenu 26 jako „generující instanci" kořen 19 zdrojového stromu 12, $Ei(R_Z):=R_Q$; srovnej obrázek 13

2. pro každou „novou instanci" I_Z 11 cílového stromu 10

I. vypočítej hodnoty všech atributů od I_Z 11, srovnej obrázek 14

atributy z I_Z 11 se určují pomocí $Att(I_Z)$

pro všechny a ∈ $Att(I_Z)$ pomocí $Calc(a,I_Z)$ určit předpis výpočtu ar a použít na atributy cesty ke „generující instanci" 13 $Path(Ei(I_Z))$ ve zdrojovém stromu 12 jakož i na již vypočítané atributy v cílovém stromu 10 $Path(I_Z)$

(další atributy, libovolně uložené ve zdrojovém stromu 12, se mohou najít a spojit se zvláštními funkcemi)

II. vytvoř všechny následníky 14 od I_Z 11, srovnej obrázek 15

urči všechny možné třídy následníků od I_Z 11 pomocí $Nf(Obj(I_Z))$

pro všechny n ∈ $Nf(Obj(I_Z))$ pomocí Gen(n) určit pravidlo generování er a použít, když je třeba:

„generující instanci" přidělit Ei(n):=er(n, Path(Ei(I_z) ∪Sub(Ei(I_z))) (zvláštními funkcemi se mohou také zde zjišťovat „generující instance", které jsou uložené libovolně ve zdrojovém stromu)
n se stane „novou instancí"

IZ 11 se stane „starou instancí", (obrázek 16)

Algoritmus je uzavřen, když všechny uzly 11, 14 atd. cílového stromu jsou „staré" uzly. Tzn., že se už nemohou vytvářet žádné „nové" uzly jakc následníci již vytvořených instancí cílového stromu a všechny atributy jsou vypočítány.

S konstrukcí cílového stromu 10 je ukončena hlavní práce konvertoru 1. Vytvořený cílový strom 10 je možné ještě vhodným způsobem dát k dispozici Backend konvertoru k dalšímu zpracování. V realizovaném řešení to znamená k Delta - generování s daty předchozího dne a k přenosu rozdílu na TMN-platformu ve tvaru jednotlivých sdělení.

Zastupuje:
Dr. Miloš Všetečka v.r.

JUDr. Miloš Všetečka
advokát
120 00 Praha 2, Hálkova 2

PATENTOVÉ NÁROKY

1. Způsob konverze stromově strukturovaných dat libovolného zdrojového objektového modelu, daných k dispozici prvním zařízením pro zpracování dat, na data volně definovatelného cílového objektového modelu, která se nechají číst druhým zařízením pro zpracování dat, pomocí nezávislého regulačního systému, přičemž zdrojový a cílový strom se skládají z objektů jedné nebo několika tříd objektů, které obsahují vždy jeden nebo několik atributů, ***vyznačující se*** následujícími kroky

a) vytvoření kořenu R_Z cílového stromu a označení kořenu zdrojového stromu jako generující instance pro kořen cílového stromu;

b) kořen cílového stromu označit jako „nový";

c) zjištění všech pravidel k vyhledávání podle výpočetních instancí zdrojového stromu, která jsou zapotřebí k výpočtu atributů nového uzlu cílového stromu (pravidla okolí);

d) stanovení výpočetních instancí použitím pravidel okolí na generující instanci uzlu zdrojového stromu;

e) zjištění všech pravidel, patřících nově vytvořenému uzlu cílového stromu, k výpočtu atributů tohoto uzlu;

f) zjištění hodnot atributů nového uzlu cílového stromu použitím pravidel výpočtu atributů na výpočetní instance a na generující instanci;

g) zjištění všech pravidel ke generování následníků uzlu cílového stromu;

h) generování následníků použitím pravidel generování na generující instanci uzlu zdrojového stromu, přitom označit příslušný uzel zdrojového stromu jako

generující instanci pro uzel následníka;

i) nový uzel cílového stromu označit jako „starý" a všechny jeho následníky označit jako „nový";

j) zvolit nový uzel cílového stromu a opakování postupu od kroku c), až není k dispozici žádný nový uzel.

2. Způsob podle nároku 1, ***vyznačující se tím***, že vstupní data se předběžně zpracovávají analytickým algoritmem a dávají se k dispozici v pevném formátu v souboru.

3. Způsob podle nároku 1 nebo 2, ***vyznačující se tím***, že konvertovaná data se dávají k dispozici v pevném formátu v souboru pro další zpracování.

4. Způsob podle některého z nároků 1 až 3, ***vyznačující se tím***, že jako kořen cílového stromu a kořen zdrojového stromu se označují ty třídy objektů, které leží v hierarchii tříd výše, než všechny ostatní třídy objektů.

5. Způsob podle některého z nároků 1 až 4, ***vyznačující se tím***, že pro konstrukci nového dílčího stromu pod pozorovaným uzlem cílového stromu se pozoruje jenom dílčí strom generující instance, přiřazené tomuto uzlu cílového stromu.

6. Způsob podle některého z nároků 1 až 5, ***vyznačující se tím***, že pro každou třídu objektů cílového stromu je definováno přesně jedno pravidlo generování.

7. Způsob podle některého z nároků 1 až 6,

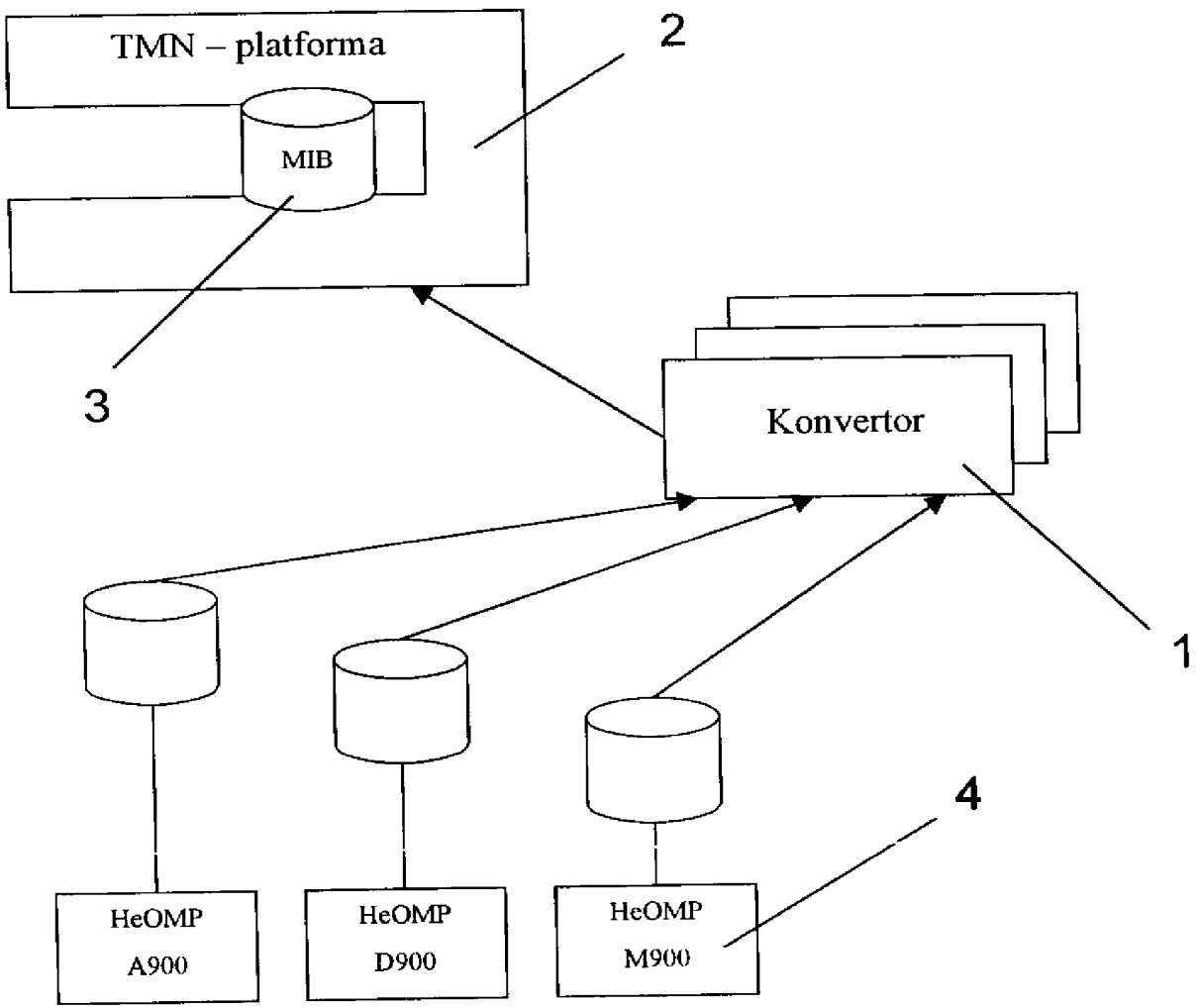
vyznačující se tím, že podle pravidla generování se zdrojový strom v okolí generující instance prohledává po instancích určité třídy objektů, které mají splňovat speciální podmínky, a při úspěšném vyhledávání se v cílovém stromu vytváří nový uzel a vytváří se instance zdrojového stromu k příslušné generující instanci uzlu cílového stromu.

8. Způsob podle některého z nároků 1 až 7, ***vyznačující se tím***, že se vytváří uzel cílového stromu jenom když se s pomocí příslušného pravidla generování nechá pro něj ve zdrojovém stromu zjistit generující instance.

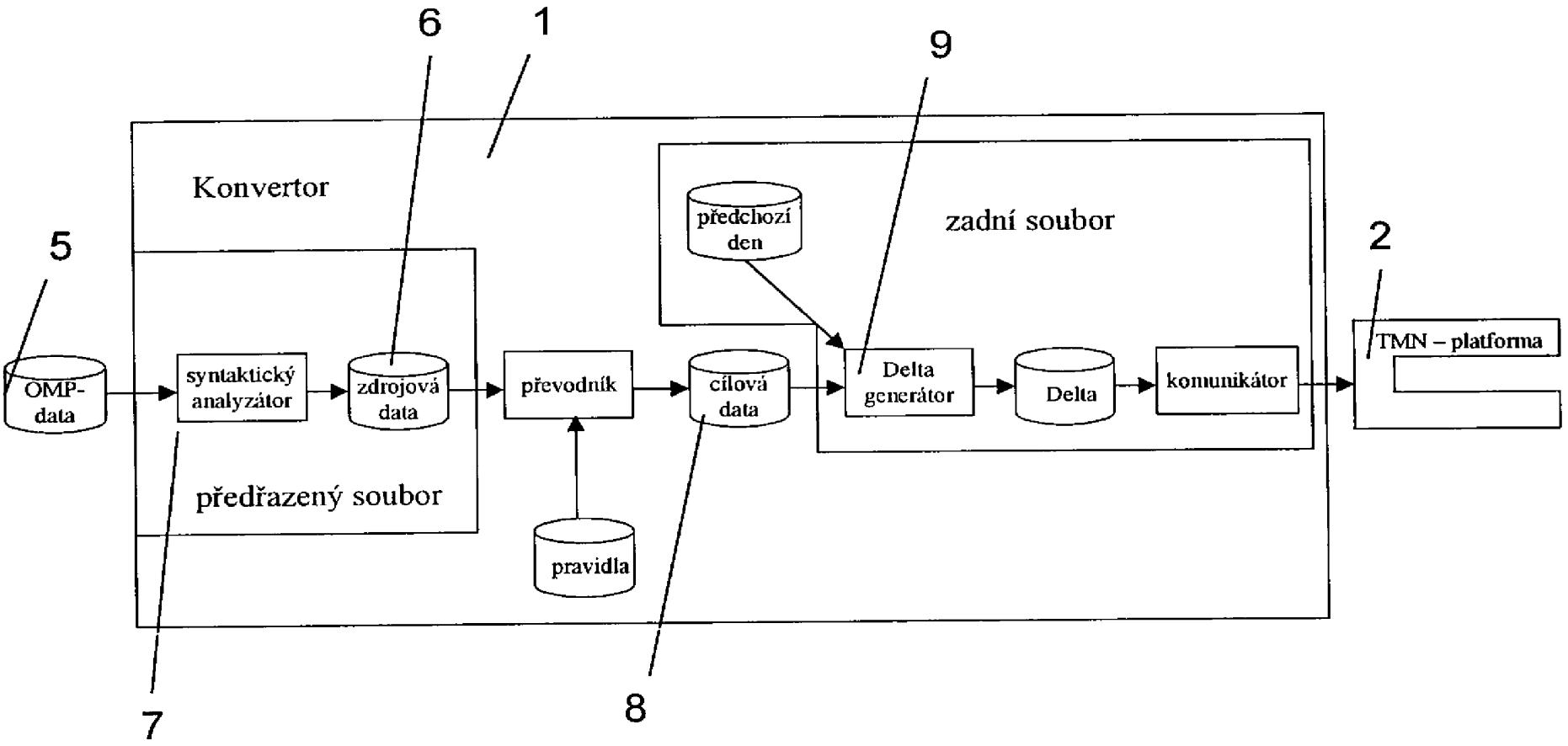
9. Způsob podle některého z nároků 1 až 8, ***vyznačující se tím***, že pravidla k výpočtu atributů jsou funkce, podle kterých se z atributů různých objektů zdrojového stromu počítají atributy nového uzlu cílového stromu.

10. Způsob podle některého z nároků 1 až 9, ***vyznačující se tím***, že přípustné třídy objektů, přípustní následníci jednotlivých tříd objektů a atributy, příslušné ke každé třídě objektů, se připravují v regulačním mechanizmu.

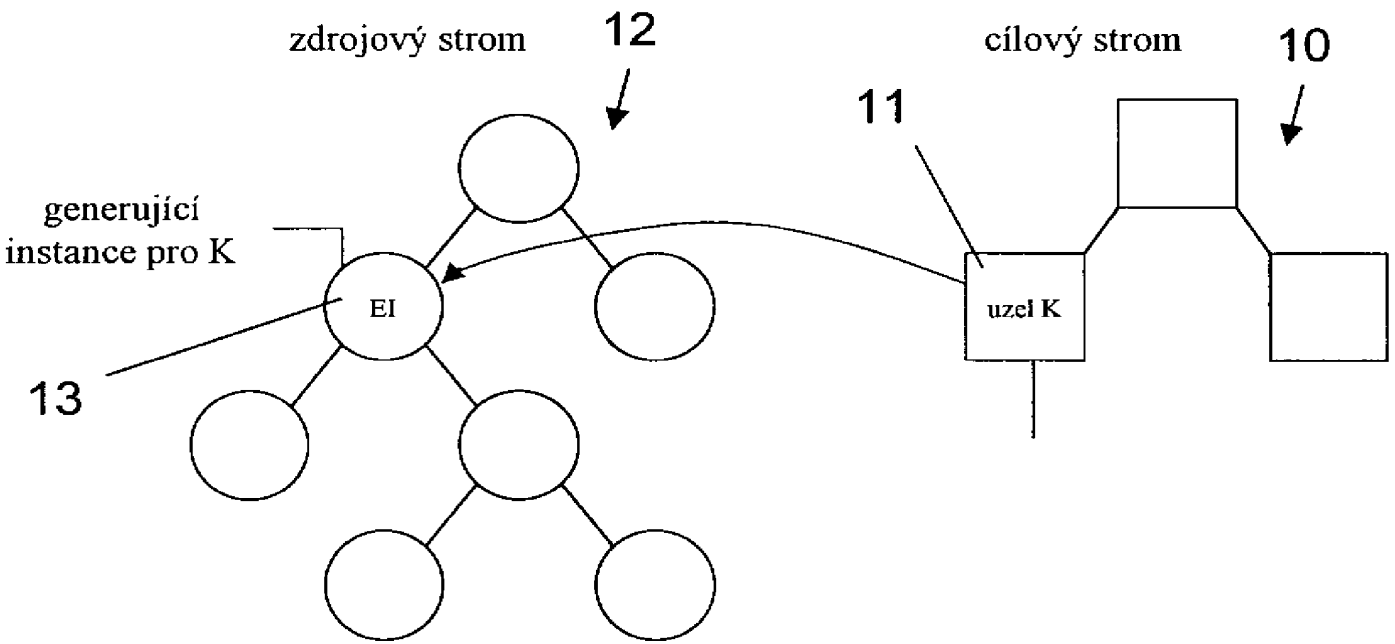
Zastupuje:
Dr. Miloš Všetečka v.r.


TMN – platforma
MIB
2
3
Konvertor
1
4
HeOMP
A900
HeOMP
D900
HeOMP
M900

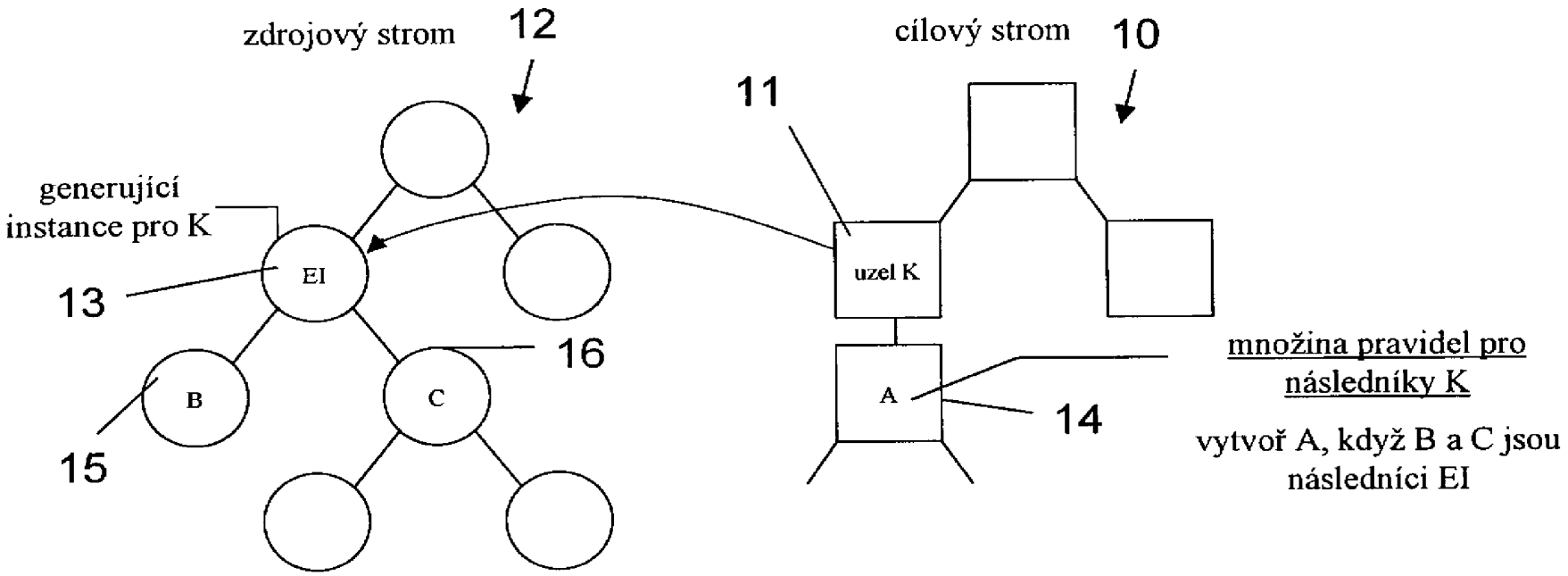
Obr. 1


6
1
9
Konvertor
předchozí den
zadní soubor
5
2
OMP-data
syntaktický analyzátor
zdrojová data
převodník
cílová data
Delta generátor
Delta
komunikátor
TMN – platforma
předřazený soubor
pravidla
7
8

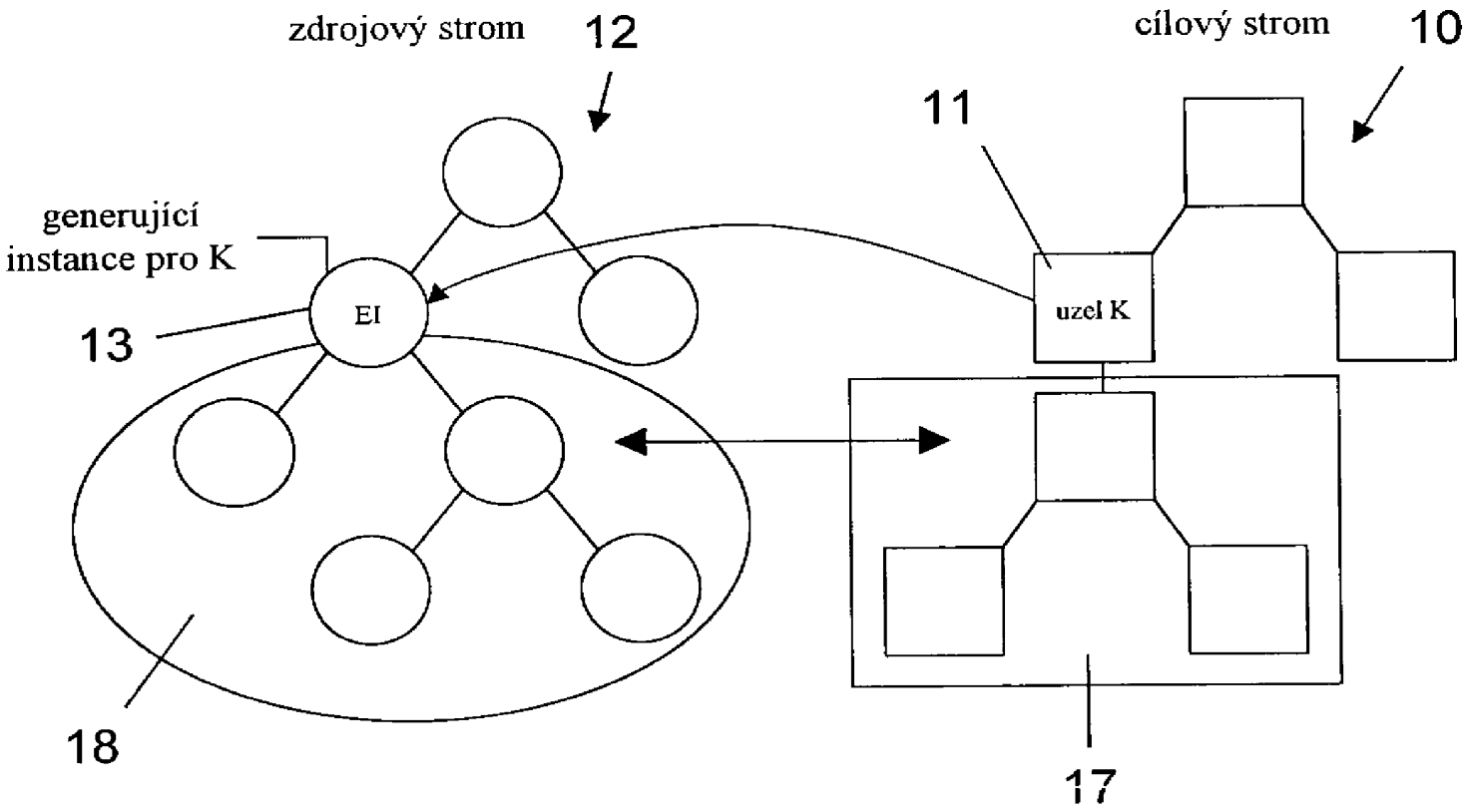
Obr. 2


zdrojový strom
12
cílový strom
10
11
generující
instance pro K
EI
uzel K
13

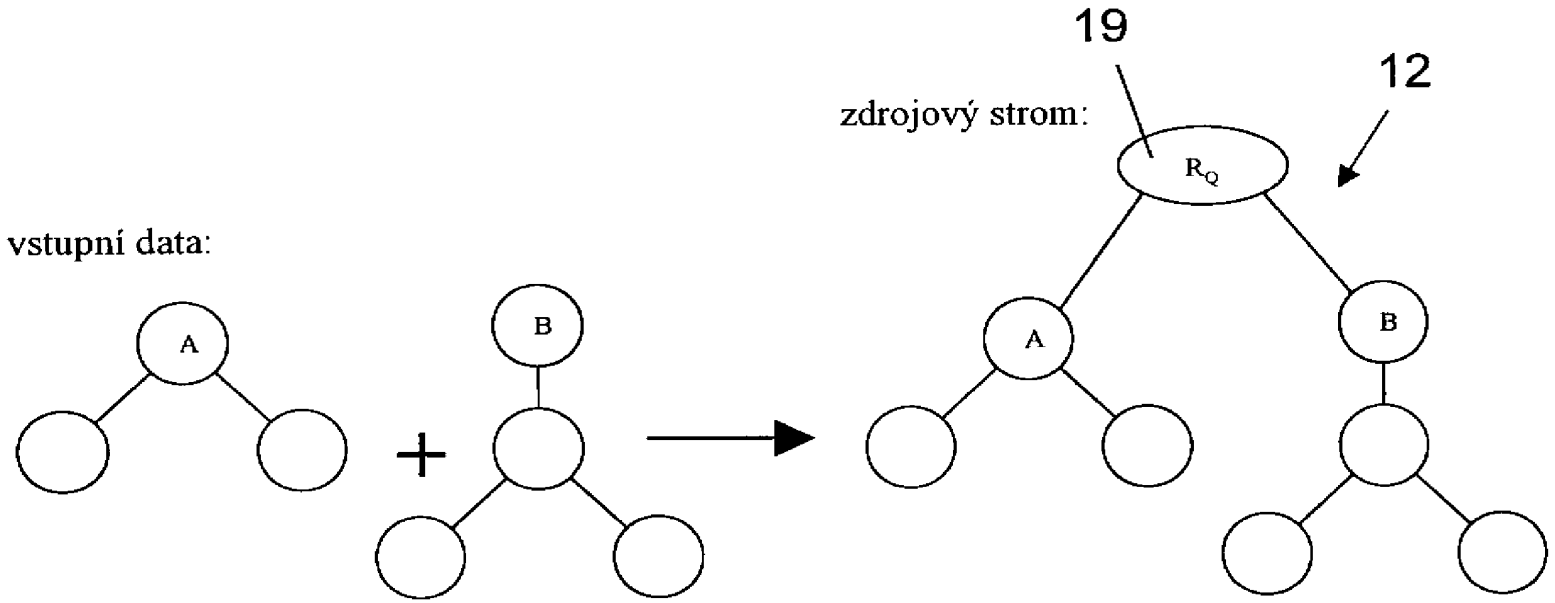
Obr. 3


zdrojový strom
12
cílový strom
10
11
generující
instance pro K
13
EI
uzel K
16
B
C
A
15
14
množina pravidel pro
následníky K
vytvoř A, když B a C jsou
následníci EI

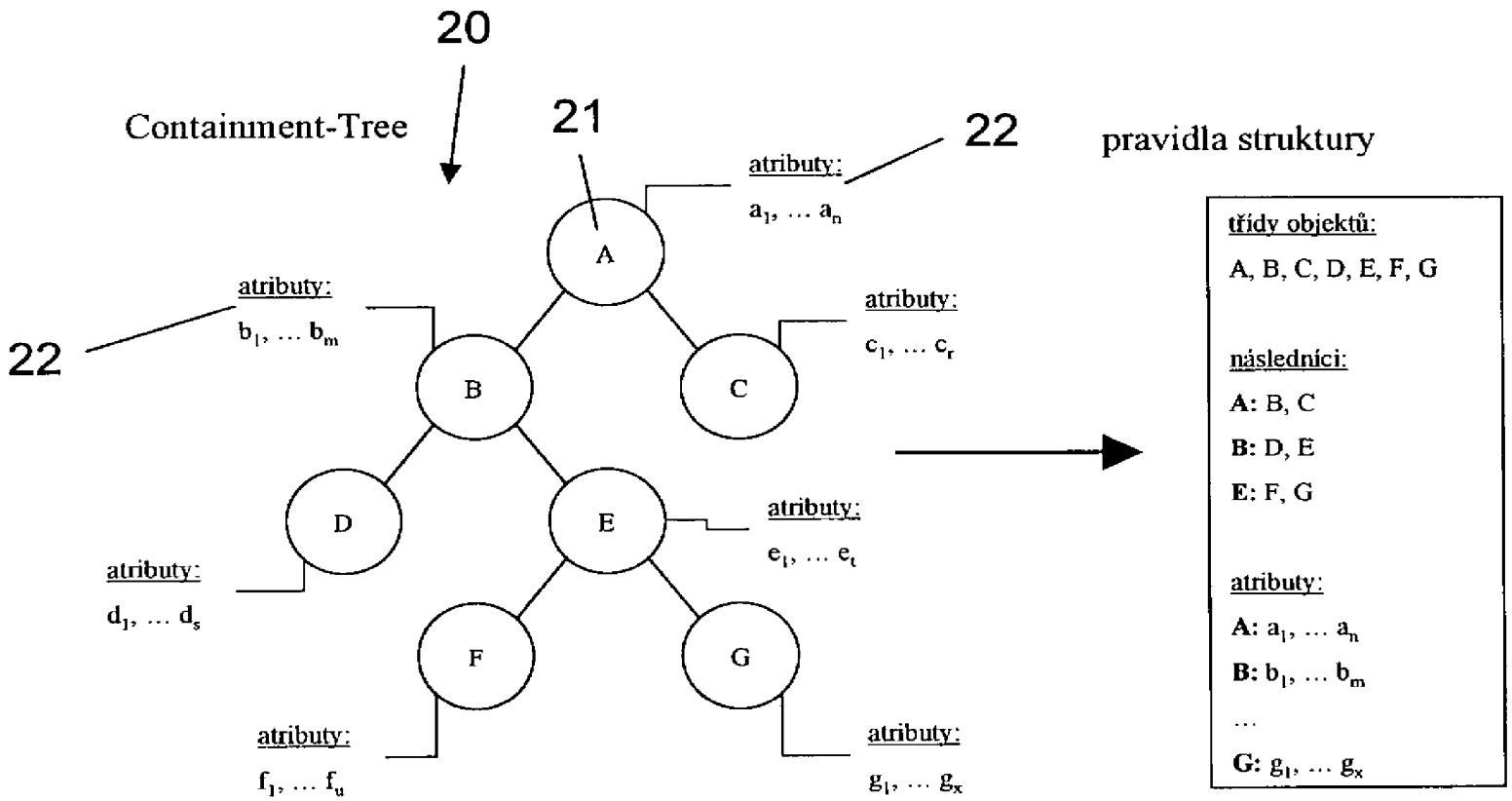
Obr. 4


zdrojový strom
12
cílový strom
10
11
generující
instance pro K
EI
13
uzel K
18
17

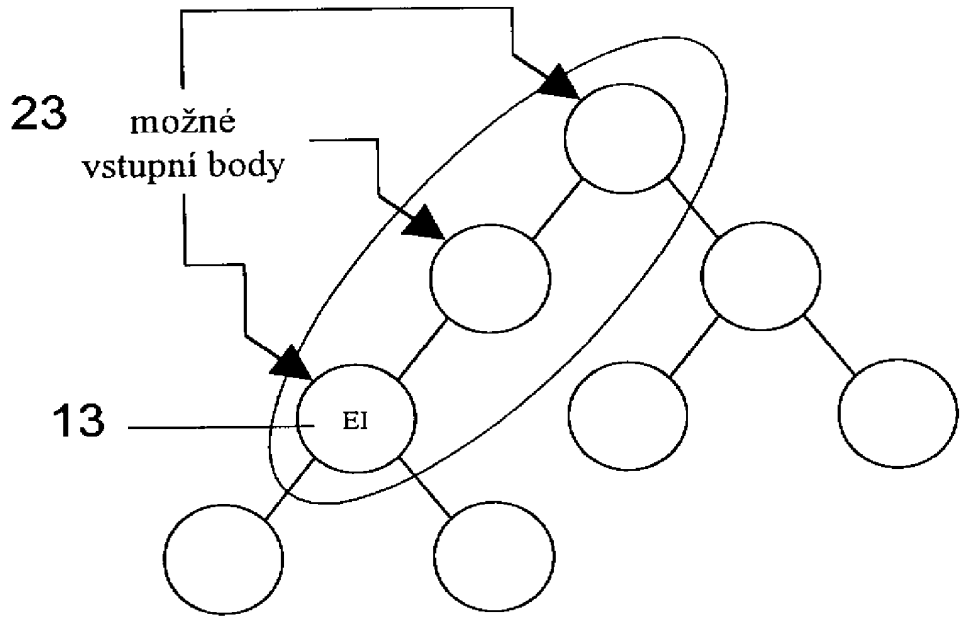
Obr. 5



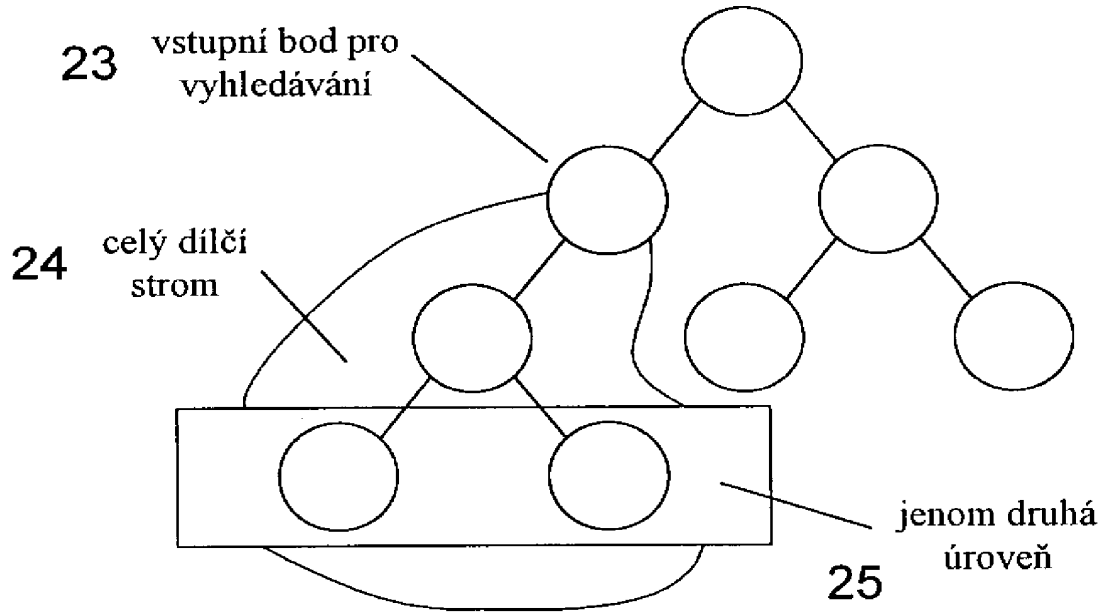
Obr. 6


20
Containment-Tree
21
22
pravidla struktury
atributy:
$a_1, \ldots a_n$
A
22
atributy:
$b_1, \ldots b_m$
B
C
atributy:
$c_1, \ldots c_r$
D
E
atributy:
$e_1, \ldots e_t$
atributy:
$d_1, \ldots d_s$
F
G
atributy:
$f_1, \ldots f_u$
atributy:
$g_1, \ldots g_x$
třídy objektů:
A, B, C, D, E, F, G
následníci:
A: B, C
B: D, E
E: F, G
atributy:
A: $a_1, \ldots a_n$
B: $b_1, \ldots b_m$
...
G: $g_1, \ldots g_x$

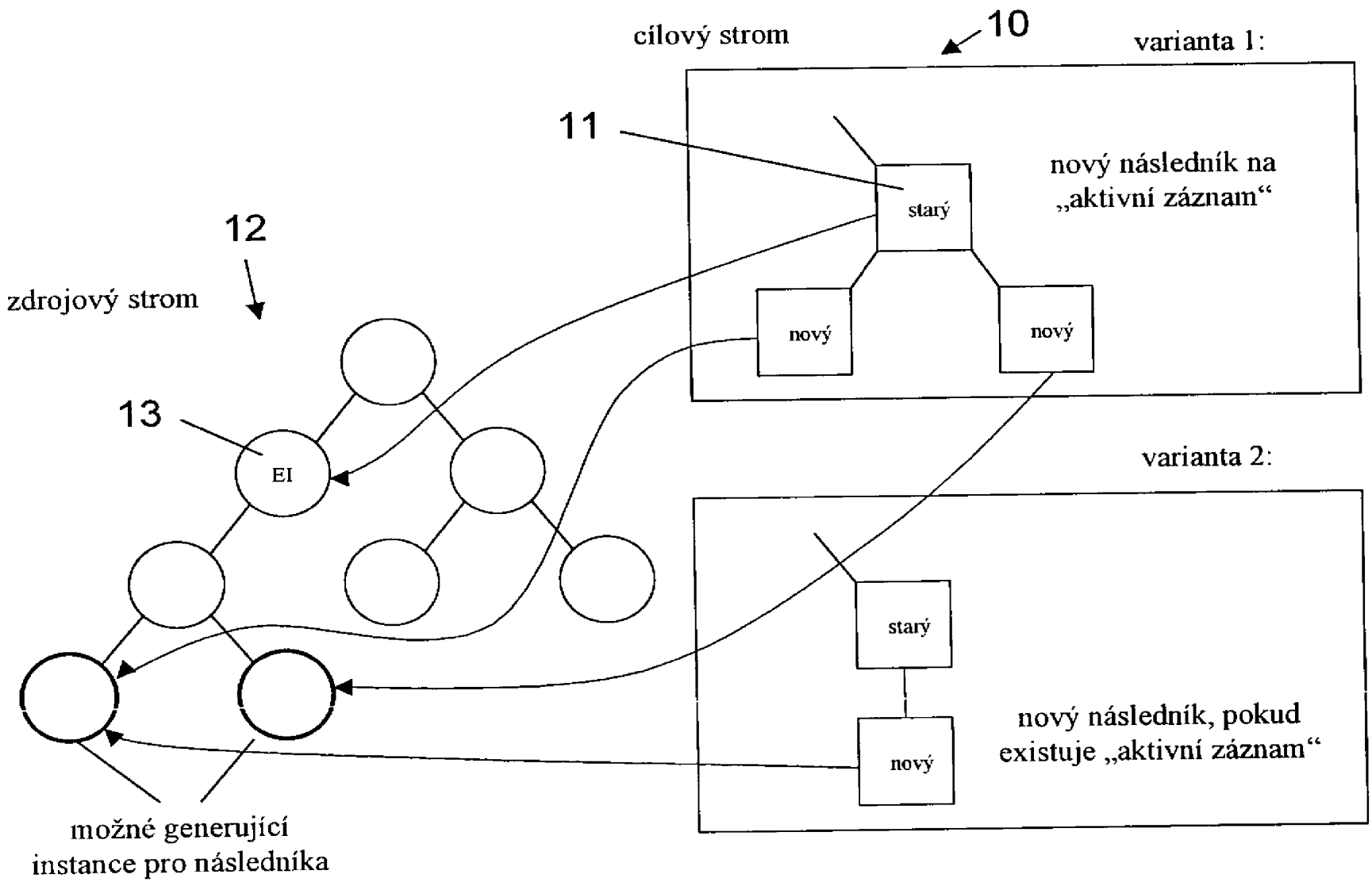
Obr. 7



Obr. 8

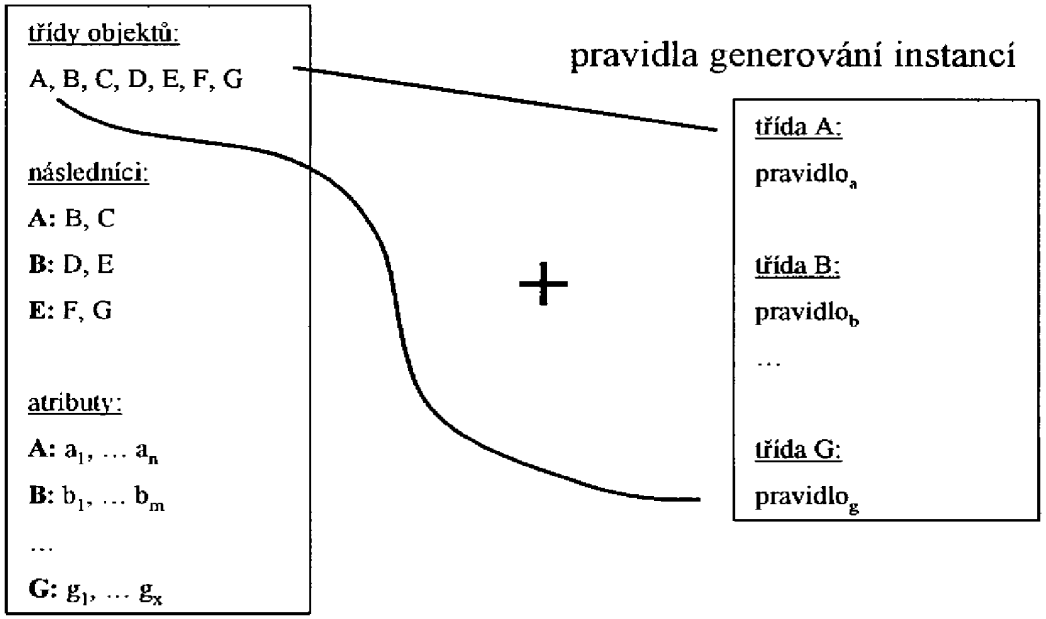

23
vstupní bod pro vyhledávání
24
celý dílčí strom
jenom druhá úroveň
25

Obr. 9

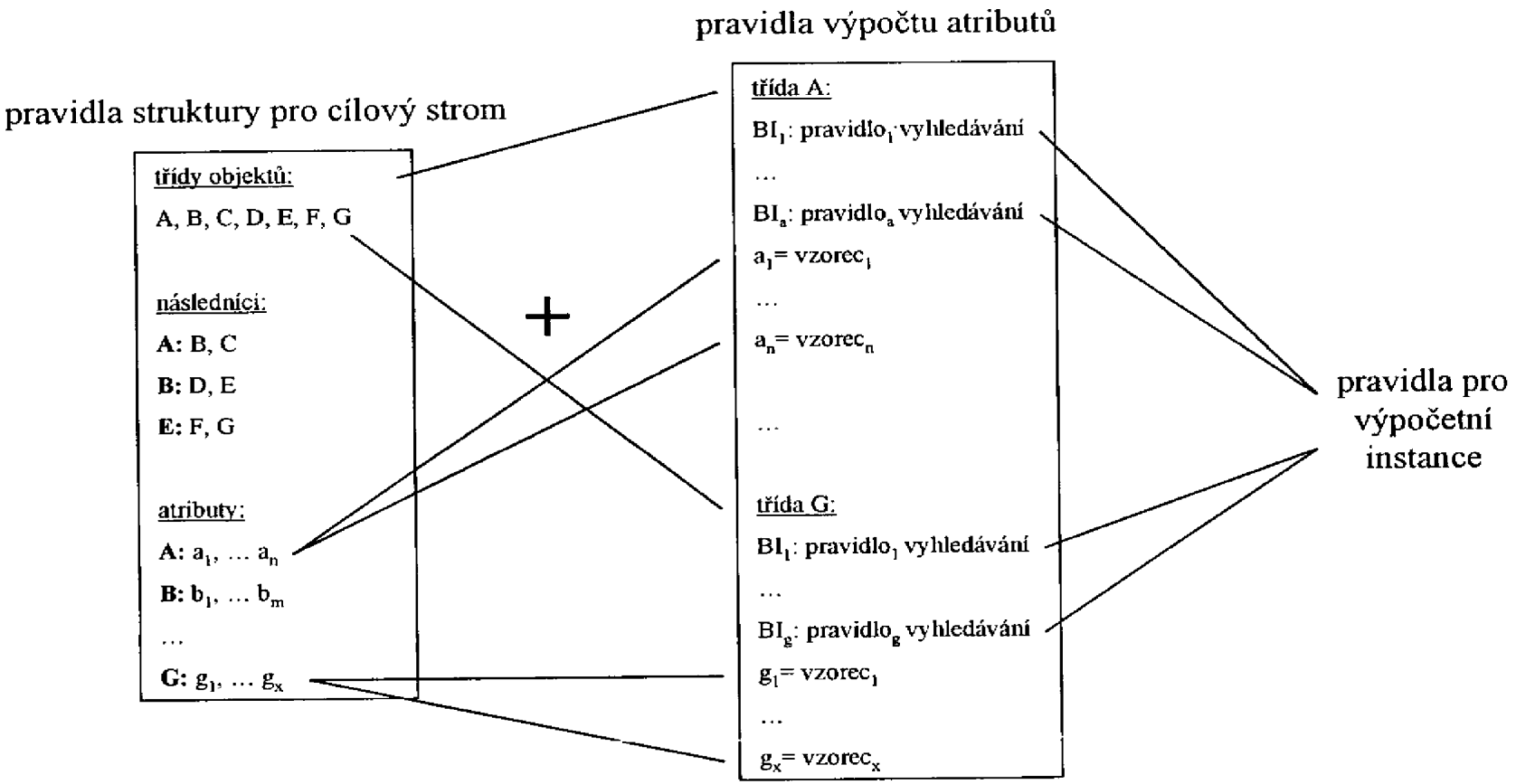

cílový strom
10
varianta 1:
11
nový následník na
„aktivní záznam“
starý
nový
nový
12
zdrojový strom
13
EI
varianta 2:
starý
nový
nový následník, pokud
existuje „aktivní záznam“
možné generující
instance pro následníka

Obr. 10

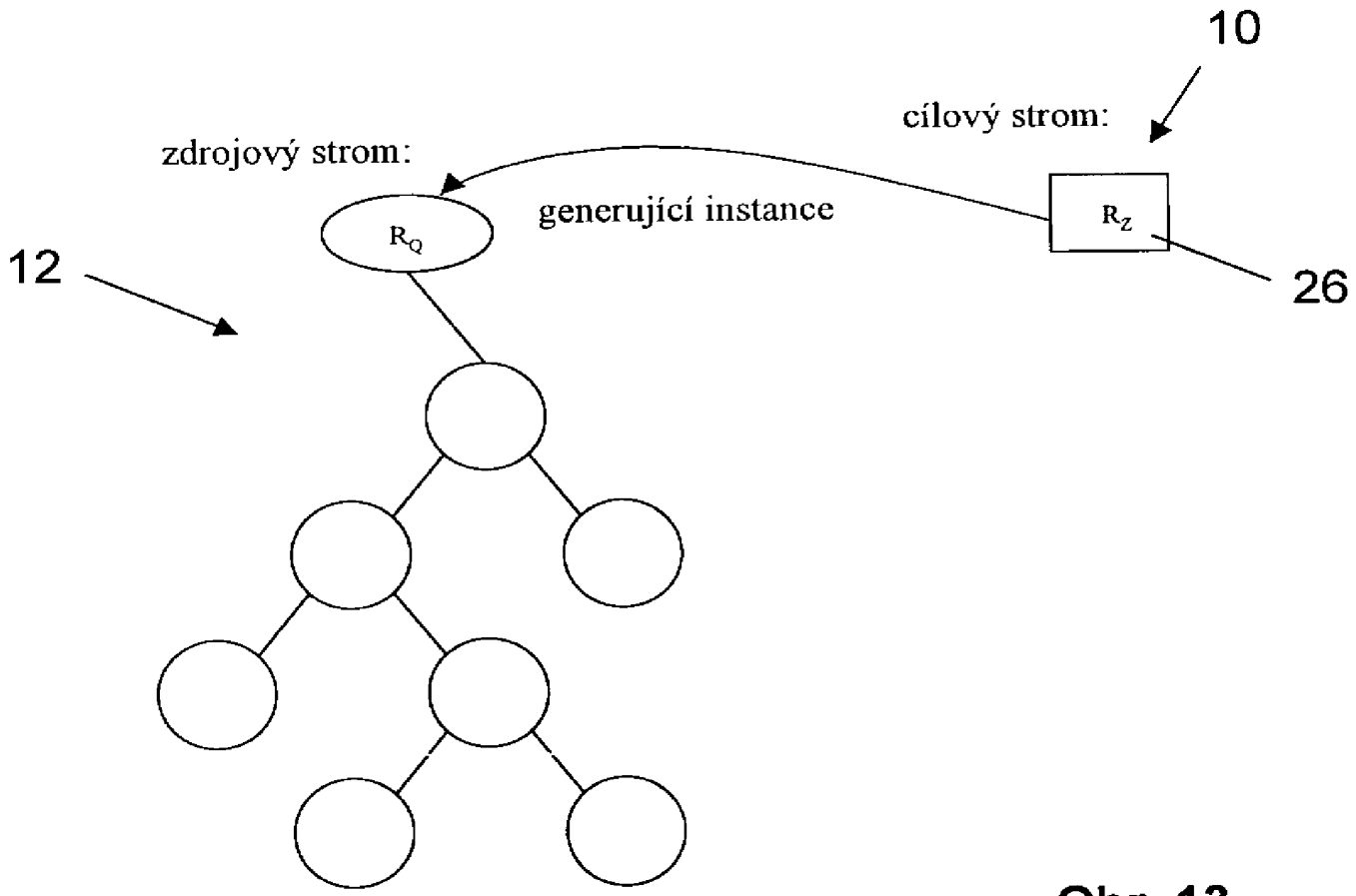
pravidla struktury pro cílový strom



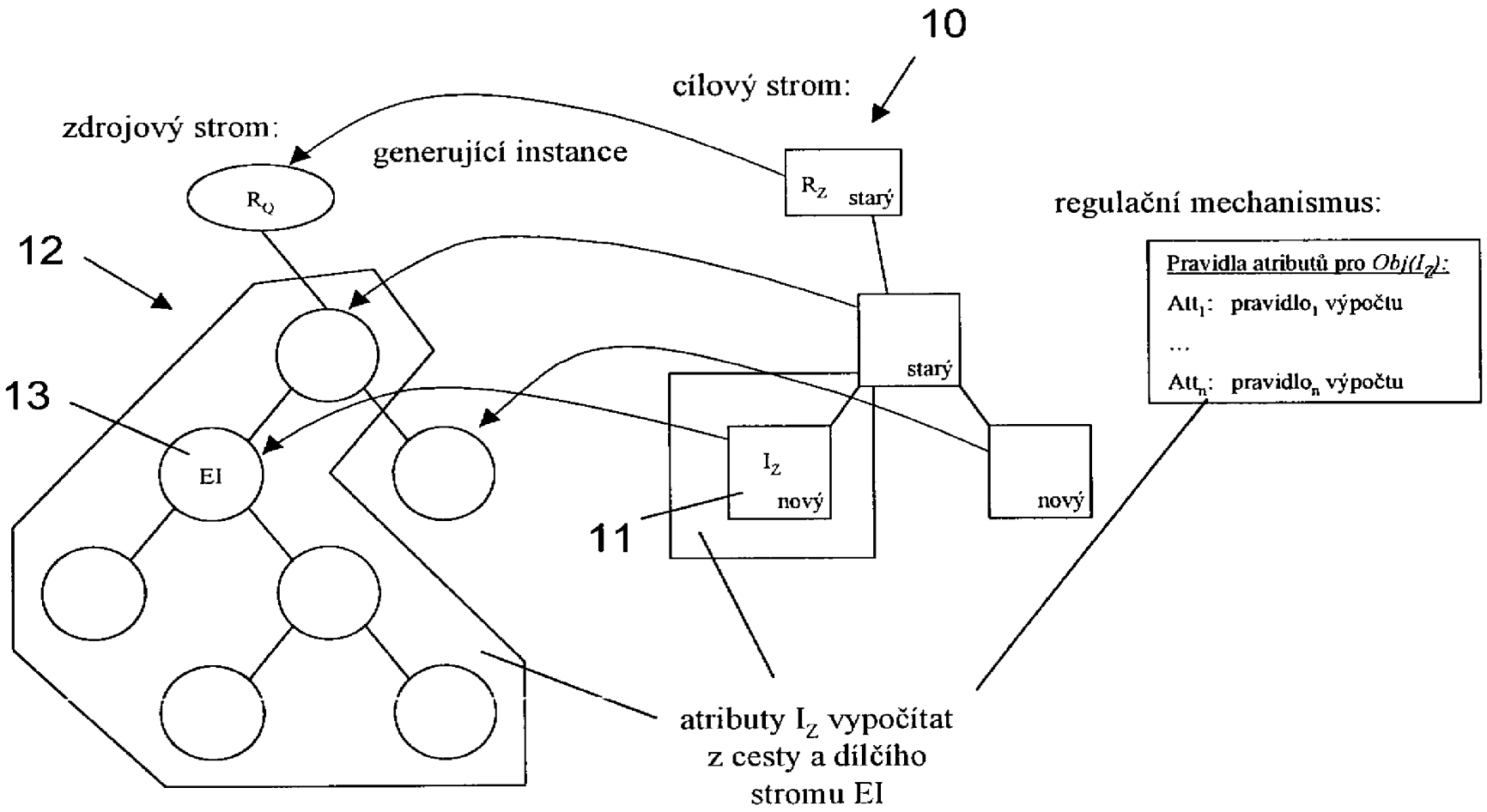
Obr. 11


pravidla výpočtu atributů
pravidla struktury pro cílový strom
třídy objektů:
A, B, C, D, E, F, G
následníci:
A: B, C
B: D, E
E: F, G
atributy:
A: a1, … an
B: b1, … bm
…
G: g1, … gx
+
třída A:
BI1: pravidlo1 vyhledávání
…
BIa: pravidloa vyhledávání
a1= vzorec1
…
an= vzorecn
…
třída G:
BI1: pravidlo1 vyhledávání
…
BIg: pravidlog vyhledávání
g1= vzorec1
…
gx= vzorecx
pravidla pro výpočetní instance

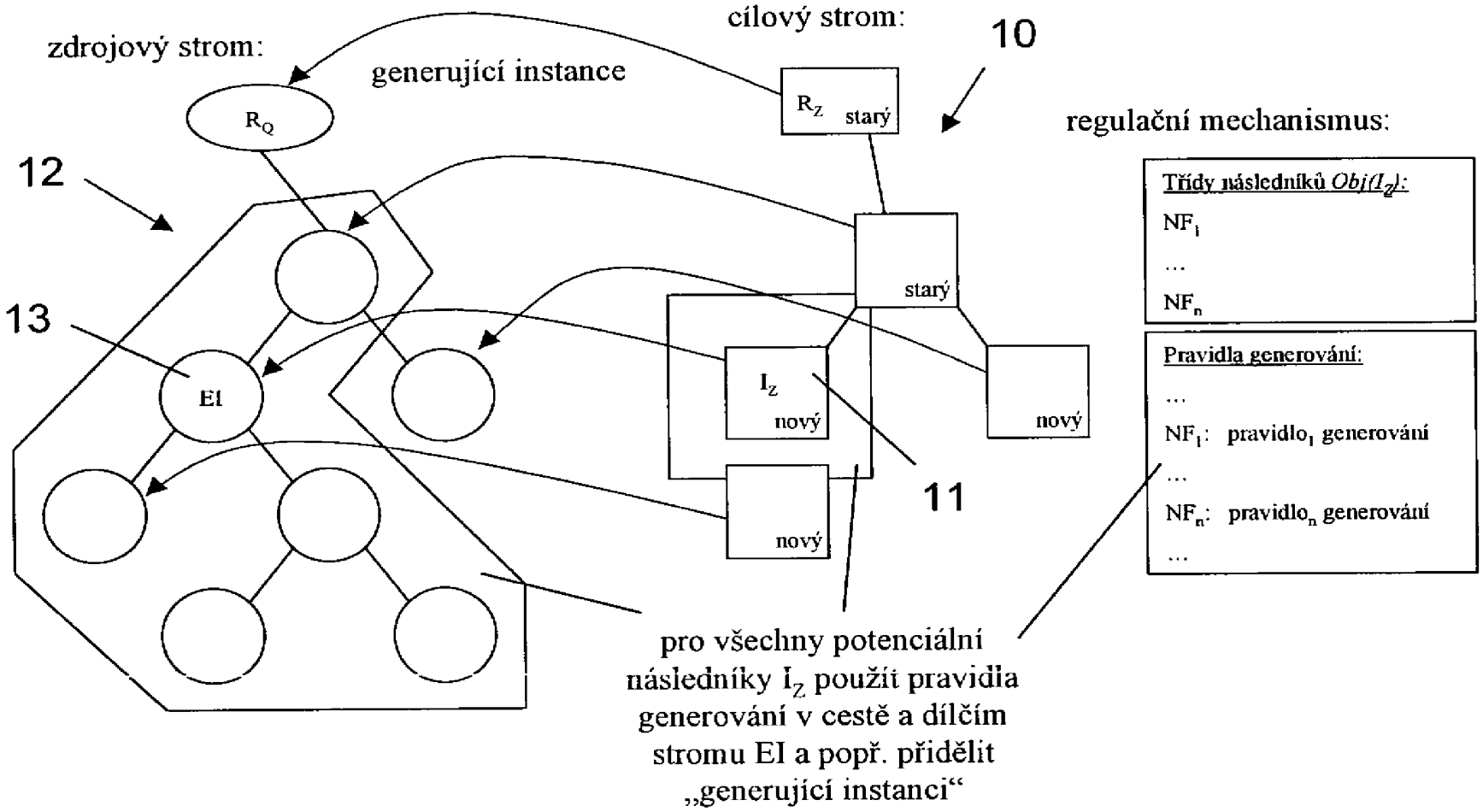
Obr. 12


10
cílový strom:
zdrojový strom:
generující instance
R_Q
R_Z
12
26

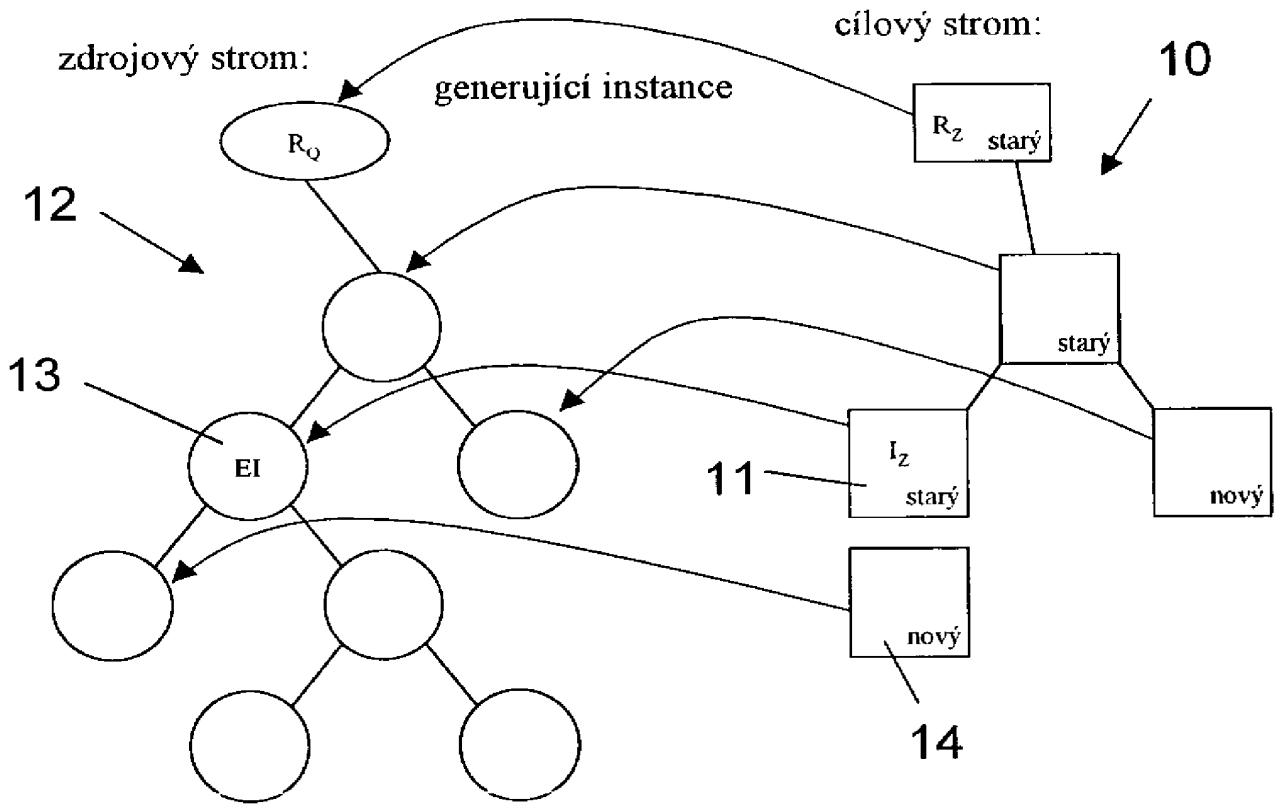
Obr. 13


10
cílový strom:
zdrojový strom:
generující instance
R_Q
R_Z starý
regulační mechanismus:
12
Pravidla atributů pro $Obj(I_Z)$:
Att_1: $pravidlo_1$ výpočtu
...
Att_n: $pravidlo_n$ výpočtu
starý
13
EI
I_Z
nový
nový
11
atributy I_Z vypočítat
z cesty a dílčího
stromu EI

Obr. 14


zdrojový strom:
generující instance
cílový strom:
10
regulační mechanismus:
R_Q
R_Z starý
12
13
EI
starý
I_Z nový
nový
nový
11
Třídy následníků $Obj(I_Z)$:
NF_1
...
NF_n
Pravidla generování:
...
NF_1: pravidlo$_1$ generování
...
NF_n: pravidlo$_n$ generování
...
pro všechny potenciální následníky I_Z použít pravidla generování v cestě a dílčím stromu EI a popř. přidělit „generující instanci“

Obr. 15


zdrojový strom:
generující instance
cílový strom:
10
R_Q
R_Z starý
12
starý
13
EI
I_Z
starý
11
nový
nový
14

Obr. 16