Statická analýza práce s daty v prostředí Vega

Kristýna Petrlíková

16.04.2024

Anotace

Vega je populární nástroj pro tvorbu grafů, který v dnešní době postrádá statickou analýzu přístupu k datům, což značně ztěžuje proces výroby validní specifikace grafu. Cílem projektu je poskytnout rozhraní, které pro danou specifikaci splňující požadavky na statičnost detekuje místa, na nichž se ve specifikaci přistupuje k neexistujícím datům.

Základní informace

Popis a zaměření ročníkového projektu

Vega je prostředí sloužící k deklarativnímu popisu statistických grafů ve formátu JSON. Specifikace grafu je částečně kontrolovaná pomocí validátoru JSON schématu, ale to je schopno vynutit pouze přítomnost, formát či typ záznamů s předem známým klíčem. Speciálně nedokáže kontrolovat hodnotu záznamu na základě hodnoty jiného záznamu nebo uživatelem specifikovaného klíče (tedy takového, jehož hodnota není zapsána v JSON schématu projektu Vega).

Zdroje dat v prostředí Vega mají jednak jejich název definovaný pomocí hodnoty konkrétního záznamu, jednak jsou samotná data popsána pomocí uživatelsky specifikovaných klíčů. Tím pádem odkazy na data a jejich atributy nelze pomocí JSON schématu kontrolovat. Zřejmě kvůli tomu, že data lze různě dynamicky načítat, Vega neposkytuje téměř žádnou kontrolu přístupu k atributům jednotlivých datových zdrojů.

Zároveň ale během runtimu uživatele neupozorní na to, že ve specifikaci přistupoval k neexistujícím atributům. To je opět dáno dynamičností systému – atributy jednotlivých datových položek mohou být kdykoliv pozměněné podle JSON specifikace (např. na základě uživatelského vstupu, jako je kliknutí myši), případně atributy existují jenom na podmnožině datových položek jednoho datového zdroje. Běžně se ovšem aspoň dá předpokládat, že data z jednoho zdroje budou mít stejnou strukturu, tedy všechny datové položky budou sdílet ty samé atributy.

Jediné, co je momentálně v prostředí Vega kontrolováno, jsou odkazy na *názvy* datových zdrojů, neboť ty musí být předem známé (avšak jejich obsah může být v podstatě libovolný).

Ročníkový projekt si klade za cíl implementovat statickou analýzu práce s daty ve specifikacích grafů v prostředí Vega. Projekt bude omezen na specifikace, které s daty pracují staticky – například se v nich nevyskytují triggery a atributy dat jsou předem známé.

Použité technologie

- Vega v5.28.0
- Vega editor
- JavaScript
- TypeScript

Odkazy (Reference)

- JSON schéma projektu Vega:
  - https://github.com/vega/schema/blob/master/vega/v5.28.0.json
- Webové stránky projektu Vega:
  - https://vega.github.io/vega/
- Githubový repozitář projektu Vega editor:
  - https://github.com/vega/editor/

Stručný popis softwarového díla

Důvod vzniku softwarového díla a jeho základní části a cíle řešení

Důvodem vzniku je doplnění neexistující funkcionality v prostředí Vega, jak bylo předesláno v předchozí části. Navíc chyby nutně nemusí být způsobeny jenom překlepy nebo neúplným přejmenováním. Kupříkladu spousta transformací vyrábí atributy s nečekanými názvy, a lze se v nich proto snadno ztratit.

Dílo bude implementováno jako modul, který na vstupu dostane validní specifikaci (tj. úspěšně zkontrolovanou JSON Schema validátorem) a na výstupu vypíše seznam chyb s jejich umístěním na řádku.

Pokud zbude čas, modul bude integrován do Vega editoru, kde by zvýrazňoval chyby přímo v textu specifikace.

Motivační příklady užití

Zde je několik příkladů situací, které v prostředí Vega momentálně nejsou ošetřené (to, že uživatel popsal neexistující data, se dozví jedině tak, že ve vykresleném grafu budou chybět).

Ve všech sekcích kromě poslední se vyskytuje následující formát chybové hlášky:

```
[Error] line `cislo_radku`, column `cislo_sloupce`:
Dataset `table_name` doesn't contain a field named `field_name`.
Available fields are `field_name1, ... , field_nameN`.
```

U jednotlivých příkladů jsou potom pro úplnost uvedena místa chyb v souborech a relevantní kontext dat.

Základy příkladů (doplněné o chyby) jsou převzaty ze sekce Examples na webové stránce projektu Vega.

Samotné příklady jsou v adresáři `examples/` přiložené k tomuto dokumentu. Níže se na ně pouze odkazujeme skrz názvy souborů.

Přístupy k datům bez transformací

Příklad 1: Doména škály

Příklad 1.1: Jeden zdroj, jeden atribut (`01_scale-domain_one-source-one-field.vg.json`)

Chyba je na řádku 27. Datový zdroj `table` obsahuje atributy `category` a `amount` (nikoliv však `cat`).

Příklad 1.2: Jeden zdroj, více atributů (`01_scale-domain_one-source-multiple-fields.vg.json`)

Chyba je na řádku 92. Datový zdroj `summary` obsahuje atributy `variety`, `mean`, `stdev`, `stderr`, ..., `stdev0`, `stdev1`, `stderr0`, `stderr1` (nikoliv však `stdev2`).

Příklad 1.3: Parametry třídění domény (`01_scale-domain_sorting.vg.json`)

Chyba je na řádku 85. Datový zdroj je stejný jako v příkladu 1.2, a neobsahuje atribut `mean0`.

Přístup k transformovaným datům

Příklad 2: Agregační transformace

Příklad 2.1: Implicitní názvy transformovaných atributů (`02_aggregate_implicit_outputs.vg.json`)

Chyba je na řádku 76. Datový zdroj `tuples` má atributy `a`, `b` a `average_c`, nikoliv však `c`.

Příklad 2.2: Explicitní názvy transformovaných atributů (`02_aggregate_explicit_outputs.vg.json`)

Chyba je na řádku 77. Datový zdroj `tuples` má atributy `a`, `b` a `d`, nikoliv však `c`.

Příklad 2.3: Implicitní operace `count` (`02_aggregate_implicit_ops.vg.json`)

Chyba je na řádku 117. Z řádků 42-45 (a dokumentace agregační transformace) vyčteme, že datový zdroj `summary` má definovaný atribut `count`, nikoliv `Count`.

Příklad 3: Zásobníková transformace (stack transform)

Příklad 3.1: Implicitní názvy transformovaných atributů (`03_stack-implicit-names.vg.json`)

Chyba je na řádku 83. Datový zdroj `table` obsahuje atributy `key`, `value` a také `y0` a `y1` (ze zásobníkové transformace), nikoliv však `y2`.

Příklad 3.2: Explicitní názvy transformovaných atributů (`03_stack-explicit-names.vg.json`)

Chyba je na řádku 108. Datový zdroj `table` obsahuje atributy `key`, `value` a také `y1` a `y2` (ze zásobníkové transformace), nikoliv však `y0`.

Lokální kontext

Příklad 4: Skupiny značek (group marks) (`04_mark-group.vg.json`)

Chyba je na řádku 68. V lokální skupině značek existuje datový zdroj `table` s atributy `x`, `y`, `w`, `h`, nikoliv však `width`.

Příklad 5: Reaktivní geometrie – značky vytvořené z jiných značek dostupných v lokálním kontextu (`05_reactive-geometry.vg.json`)

Chyba je na řádku 117. Množina značek definovaná na řádcích 111-120 vzniká z atributů množiny značek `layout`, jenže ta má mj. atributy `opacity`, `x`, `y`, `fill`, nikoliv však `file`.

Transformace s výrazy (expressions)

Příklad 6.1: Transformace pomocí formule (formula transform) (`06_formula-transform.vg.json`)

Chyba je ve výrazu na řádku 59. Odkazované `datum`, které je datovou položku ve zdroji `trellis`, obsahuje atributy `a`, `count span`, `y0` a `y1`, nikoliv však `Count`.

Příklad 6.2: Filtrační transformace (filter transform) (`06_filter-transform.vg.json`)

Chyba je ve výrazu na řádku 16. Datový zdroj `source` před transformací obsahoval atributy `Name`, `Horsepower`, atd., nikoliv však `Horspower`.

Netabulková data

Chybové hlášky v této části mají následující formát:

```
[Error] line `cislo_radku`, column `cislo_sloupce`:
Not all data objects in `table_name` contain a field named `field_name`.
Available valid common fields are `field_name1, ... , field_nameN`.
```

Příklad 7 (`07_nontabular-data.vg.json`)

Chyba se projeví na řádku 26. Datový zdroj `source` má jediné úplné atributy `y` a `c`, nikoliv však `x`.

Time-line & Milestones

Datum	Milník	Způsob prezentace
26.04.	Finální verze specifikace	osobně
30.05.	Založení repozitáře; integrace nově vznikajícího projektu s prostředím Vega; definitivní identifikace všech míst, kde se k datům přistupuje a jak se s nimi manipuluje	osobně/v repozitáři
30.06.	Implementace řešení základních příkladů	osobně/v repozitáři
30.07.	Implementace řešení příkladů s transformovanými daty	osobně/v repozitáři
30.08.	Implementace pokročilých částí, příp. integrace do Vega editoru	osobně/v repozitáři