Domácí úkol 12: CODENAME Wi-Fi

Poslední domácí úkol budete implementovat v Prologu. Jako u předchozích velkých úkolů, i za tento můžete získat až dva body do hodnocení.

(Wi-)FI vás potřebuje

Blíží se konec semestru a s ním i zkouškové období, po jehož konci je naplánována výměna celé infrastruktury Wi-Fi v budově Fakulty informatiky z důvodu blížícího se konce životnosti současného technického vybavení. Jelikož budou Vánoce a i zaměstnanci CVT si i přes nový IS zaslouží odpočinek, tak je přiřazení kanálů[1] k jednotlivým přístupovým bodům na vás.

Takovéto přiřazování kanálů není vůbec jednoduchá záležitost, jelikož je třeba dávat si veliký pozor na to, aby se jednotlivé přístupové body se stejnými kanály nepřekrývaly. V opačném případě by mohlo docházet k negativnímu ovlivňování výkonu sítě, což by nepřímo mohlo postihnout i výkon pilně studujících studentů (ať už na přednáškách, cvičeních, v knihovně, studovně nebo v počítačové hale).

Vaším úkolem je tedy pomoci naplánovat přiřazení kanálů k jednotlivým přístupovým bodům.

Kostra řešení

Ze studijních materiálů si stáhněte kostru řešení. K dispozici máte zejména níže zmíněné predikáty.

- `ac/2`, jímž modelujeme fakt, že signál přístupového bodu `A` překrývá signál přístupového bodu `B` (ne však opačně!). Jedná se tedy o *jednosměrný* vztah. Pokud si budete chtít přidat (třeba pro účely testování) vlastní přístupové body a překryvy mezi nimi, stačí vám přidat příslušné fakty `ac(bodA, bodB)`. Přidávejte vždy **jen jeden směr** překryvu; ten druhý řeší následující predikát.
- `acOverlap/2`, který „zobousměrňuje“ `ac/2`. Jinými slovy `acOverlap(A, B)` uspěje, pokud signál přístupového bodu `A` překrývá signál přístupového bodu `B`, nebo naopak. V úloze nepočítáme s existencí jednosměrného překryvu dvou přístupových bodů.
- `accessPoints/1`, jenž do volné proměnné unifikuje seznam všech přístupových bodů. Tento seznam se získává prostřednictvím predikátu `acOverlap/2`, bude proto fungovat i tehdy, budete-li si přidávat vlastní přístupové body.
- `countSolutions/2`, který pro zadané `N` v prvním argumentu a aktuální infrastrukturu v `acOverlap/2` unifikuje do volné proměnné počet řešení. Tento predikát se vám bude hodit zejména při testování.

Zadání

Naprogramujte predikát `channelise/2`, který do proměnné v prvním argumentu unifikuje všechna čísla `N`, pro která nebude docházet k překryvům signálů přístupových bodů na stejných kanálech. Do druhé proměnné v druhém argumentu posléze unifikuje seznam dvojic takový, že platí:

- První složkou každé dvojice je název přístupového bodu, druhou složkou je číslo kanálu, který byl tomuto přístupovému bodu přiřazen, a na kterém tedy bude tento přístupový bod vysílat.
- Můžete předpokládat, že `N` je z množiny přirozených čísel s nulou.
- Kanály jsou číslované od `0` do `N - 1` včetně.
- Seznam musí obsahovat pro každý existující přístupový bod právě jednu dvojici, na jejich pořadí nezáleží.
- Pokud se signály přístupových bodů překrývají, tak nesmí mít tyto body ve výsledném seznamu přiřazeno stejné číslo kanálu.

[1] https://en.wikipedia.org/wiki/List_of_WLAN_channels

Existuje-li vícero takovýchto seznamů (zanedbáme-li jen různé pořadí dvojic), musí predikát postupně (tj. ve `swipl` po stisknutí středníku) objevit každý z nich. Seznamy, které se liší pouze pořadím dvojic, jsou z pohledu zadání stejné, vždy tedy vypište právě jeden z těchto seznamů.

Predikát má být použitelný v módu `channelise(?N, -CS)`; měl by tedy nejen dokázat najít vhodná přiřazení kanálů pro volné `N`, ale měl by také umět najít vhodná přiřazení i pro konkrétní hodnotu `N`. Pakliže takový seznam neexistuje, predikát neuspěje (tj. `swipl` vypíše na dotaz `false.`) a žádná Wi-Fi nebude.

Váš predikát musí fungovat obecně, tedy na jakékoli síťové infrastruktuře popsané predikátem `ac/2`. Rozhodně nebude uznáno řešení, jež v sobě bude mít natvrdo naprogramována všechna přiřazení kanálů vyhovující zadání.

Tipy a triky

- Pro snazší implementaci této úlohy se vám budou hodit dvojice v Prologu. Jejich zápis je de facto totožný tomu, který již moc dobře znáte z Haskellu. Následující úryvek kódu ukazuje vše, co budete k vyřešení této úlohy potřebovat.

```
X = (m, n).                    % Do X jsme unifikovali dvojici pruků m a n.
Y = [(a, b), (c, d), (e, f)]. % Do Y jsme unifikovali seznam dvojic.
```

- Při řešení úlohy by vám mohl být užitečný predikát `between(+Low, +High, ?Value)`[2], který můžete použít ke generování posloupnosti celých čísel na základě vstupních parametrů.
- Dbejte na to, aby vaše řešení nebylo zbytečně neefektivní. Rozhodně nebudou uznána taková řešení, která svůj výpočet nedokončí v dostatečně krátkém čase ani pro vzorové síťové infrastruktury.
- Pro zvídavé: Pokud se vám nelíbí, že ve svém řešení máte konkrétní implementaci síťové infrastruktury zadanou způsobem, který ji neumožňuje elegantně měnit, můžete si nastudovat, jak funguje predikát `call/1`. Takové řešení však vyžaduje zásah do kostry domácího úkolu. Mimo jiné je nutné upravit definici predikátu `channelise/2` na `channelise(:Net, ?N, -CS)`, kde `Net` je název predikátu popisujícího danou síťovou infrastrukturu.

Zobrazení výsledků v interpretu

Výsledkem výpočtu je seznam, který může být poměrně dlouhý (to záleží na velikosti vstupní sítě), a `swipl` ho proto nemusí zobrazit celý. Toto chování můžete obejít například použitím vestavěného predikátu `print/1`. Dotaz poté bude vypadat následovně:

```
?- channelise(N, CS), print(CS).
```

Druhá možnost, jak si od interpretu vyžádat vypsání celého seznamu, je použitelná pouze v případě, že vám `swipl` nabízí možnost se dotázat na další řešení. Potom stačí stisknout místo středníku nebo tečky klávesu `w`.

Testování

Jak bylo řečeno, vaše implementace predikátu `channelise/2` musí fungovat obecně. Pro účely testování v kostře naleznete tři menší příklady síťové infrastruktury. U každé z nich naleznete komentář, který obsahuje počet řešení pro danou infrastrukturu. Můžete si tedy ověřit, zdali vaše implementace vrací alespoň stejný počet výsledků.

Tyto vzorové sítě jsou značené predikáty `acN/2`, kde `N` je přirozené číslo. Doporučujeme dopsat si i vlastní příklady, abyste své řešení opravdu řádně otestovali. Pro testování by vám mělo stačit, za předpokladu, že budete využívat predikát `acOverlap/2` (což důrazně doporučujeme), pouze změnit v něm užívaný predikát `ac1/2` na svou konkrétní síť.

[2] http://www.swi-prolog.org/pldoc/man?predicate=between/3

Ukázka výpočtu

V následující ukázce byla použita síťová infrastruktura `ac3`. Připomeňme, že na pořadí dvojic ve výsledném seznamu nezáleží.

```
?- channelise(0, CS).
false.

?- channelise(1, CS).
false.

?- channelise(N, CS).
N = 2,
CS = [(d, 1),  (c, 0),  (b, 1),  (a, 0)] ;
N = 2,
CS = [(d, 0),  (c, 1),  (b, 0),  (a, 1)] ;
N = 3,
CS = [(d, 1),  (c, 0),  (b, 1),  (a, 0)] ;
N = 3,
CS = [(d, 2),  (c, 0),  (b, 1),  (a, 0)] ;
N = 3,
CS = [(d, 1),  (c, 2),  (b, 1),  (a, 0)] ;
...

?- channelise(3, CS).
CS = [(d, 1),  (c, 0),  (b, 1),  (a, 0)] ;
CS = [(d, 2),  (c, 0),  (b, 1),  (a, 0)] ;
CS = [(d, 1),  (c, 2),  (b, 1),  (a, 0)] ;
CS = [(d, 1),  (c, 0),  (b, 2),  (a, 0)] ;
CS = [(d, 2),  (c, 0),  (b, 2),  (a, 0)] ;
...
```

Předposlední volání predikátu `channelise/2` bude postupně vypisovat všechny možnosti pro všechna možná `N`. Naopak poslední volání postupně vypíše pouze 18 možností, které odpovídají pouze dané hodnotě `N`.

Odevzdávání a bodování

Na rozdíl od předchozích velkých úkolů budete tentokrát své dílo vkládat do odevzdávárny. Odevzdávejte jeden soubor s příponou `.pl`, který obsahuje okomentovaný zdrojový kód. Do komentářů slovně popište, jak jste při řešení příkladu postupovali, a případně také uveďte zdroje, kterými jste se inspirovali. Komentáře nemusíte psát anglicky. Čas na vypracování úkolu je do středy **12. prosince** (včetně).

Za celou úlohu můžete dostat dva body. První bod je za funkčnost kódu, druhý bod je za jeho čitelnost, smysluplnost a slovní popis. Je možné získat i desetinné body, pokud bude řešení jen částečně správně. Oba body uděluje cvičící.