MASARYKOVA UNIVERZITA
FAKULTA INFORMATIKY

Časovo závislé rozmiestnenie MSC

BAKALÁRSKA PRÁCA

Tomáš Márton

Brno, jar 2013

Prehlásenie

Prehlasujem, že táto bakalárska práca je mojím pôvodným autorským dielom, ktoré som vypracoval samostatne. Všetky zdroje, pramene a literatúru, ktoré som pri vypracovaní používal alebo z nich čerpal, v práci riadne citujem s uvedením úplného odkazu na príslušný zdroj.

Tomáš Márton

Vedúci práce: Mgr. Ľuboš Korenčiak

Poďakovanie

Rád by som poďakoval Mgr. Ľubošovi Korenčiakovi, vedúcemu tejto bakalárskej práce, za podnetné návrhy a početné konzultácie. Ďalej by som rád poďakoval Viktorovi Borzovi a Adrianovi Farmadinovi za pomoc pri integrácii kódu, celému tímu SCStudia, ako aj mojej rodine a priateľke za podporu pri štúdiu.

Zhrnutie

Sequence Chart Studio (SCStudio) je užívateľsky prívetivý nástroj na kreslenie a verifikáciu Message Sequence Charts (MSC) diagramov. Momentálne SCStudio vykresľuje MSC diagramy snažiac sa výsledný diagram čo najviac zjednodušiť a sprehľadniť. Táto práca sa zaoberá variantou tohto vykresľovania s ohľadom na časové značky. Práca prezentuje možnosti vykreslenia MSC diagramu a popisuje implementáciu najvhodnejšej varianty. Ďalej rozširuje SCStudio o algoritmus zjemňovania pre absolútne časové intervaly a dokazuje jeho korektnosť.

Kľúčové slová

Sequence Chart Studio, SCStudio, Message Sequence Chart, MSC, BMSC, zjemňovanie, časová konzistentnosť

Obsah

Úvod

V dnešnej dobe enormne rýchlo pribúdajú úlohy, ktoré sú riadené počítačom, či už v bežnom živote, alebo v priemysle. Z tohto dôvodu po celom svete vznikajú organizácie, ktoré vytvárajú štandardy a protokoly, vďaka ktorým je postup pri vykonávaní úloh uniformný a počítače spolu dokážu komunikovať. Aby sme však boli schopní overiť správnosť postupu, potrebujeme nástroje, ktoré nám umožnia verifikovať tento postup voči štandardu. V práci sa zaoberáme Message Sequence Chart (MSC), formalizmom pre popis komunikácie medzi objektami v systéme, ktorý je vhodný predovšetkým na popis sieťových protokolov. V grafickej podobe je to interakčný diagram podobný UML sekvenčnému diagramu a bol štandardizovaný konzorciom ITU v štandarde Z.120 [6]. Ďalej pracujeme so softvérovým nástrojom Sequence Chart Studio (SCStudio), ktorý poskytuje grafické rozhranie na kreslenie MSC diagramov a ich verifikáciu [3].

Cieľom tejto bakalárskej práce je naštudovať problematiku časových značiek v MSC diagramoch. Zároveň navrhnúť a implementovať variantu grafického vykresľovania, ktorá bude brať v úvahu časové značky, čo v terajšej implementácií chýbalo. Práca ďalej rozširuje SCStudio o vybrané časové algoritmy, potrebné na verifikáciu a kontrolnú sadu testov. Algoritmy časovej konzistentnosti a zjemňovania síce boli obsiahnuté v SCStudiu, ale neposkytovali možnosť pracovať aj s absolútnymi časovými značkami. Preto bola táto funkcionalita pridaná a začlenená do SCStudia. Výsledný nástroj umožní jednoduchšiu orientáciu v MSC diagrame obsahujúcom časové značky a poslúži pre plnohodnotnú prácu s oboma typmi časových značiek.

Práca sa skladá zo štyroch kapitol. V prvej kapitole sa zoznámime s formalizmom MSC a nástrojom SCStudio, presnejšie zadefinujeme zadanie a popíšeme dôvody pre implementáciu. V ďalšej kapitole si zadefinujeme presný formát vstupných dát a zanalyzujeme už naimplementované algoritmy. Tretia kapitola sa venuje samotnému vykresľovaniu MSC podľa časových značiek a prezentuje možné spôsoby vykrelenia MSC diagramu. Ďalej si predstavíme rozširujúce algoritmy, ich implementáciu a dôkaz korektnosti jedného z nich. V závere zhodnotíme, či práca spĺňa zadané požiadavky a popíšeme budúci vývoj programu.

1 Predpoklady

Message Sequence Chart (MSC) predstavuje formalizmus, pre popis komunikácie medzi jednotlivými objektami v systéme, založený na základe zasielania správ. Používa sa predovšetkým na popis sieťových protokolov. MSC definuje dve reprezentácie, textovú a grafickú. Oba typy majú rovnakú popisnú silu, ale textová reprezentácia neobsahuje niektoré, pre prácu dôležité informácie o grafickom rozložení diagramu, takže sa tým naša práca bližšie nezaoberá. MSC sa ďalej rozdeľuje na basic MSC (BMSC) a high-level MSC (HMSC). BMSC slúži pre popis konečnej komunikácie konkrétneho subsystému. HMSC je abstraktnejší ako BMSC, popisuje aj vyššie vrstvy systému a môže obsahovať viaceré BMSC diagramy [3]. V práci sa HMSC ďalej nezaoberáme.

1.1 Basic Message Sequence Chart

V nasledujúcom texte najskôr poskytneme formálnu definíciu BMSC a potom si ju na grafickej reprezentácii BMSC vysvetlíme. Nasledujúce definície sú použité z [7].

Definícia 1. *$\mathcal{I}$ je konečná množina všetkých intervalov tvaru*

$$(a, b), (a, b], [a, b), [a, b]$$

kde $a, b \in \mathbb{R} \cup (-\infty, \infty) \wedge a \leq b$.

Definícia 2. *BMSC môžme definovať ako usporiadanú 8-micu*

$$(E, <, \mathcal{P}, \tau, P, \mathcal{M}, \mathcal{K}, \mathcal{L})$$

kde:

- *E je konečná množina všetkých udalostí.*
- *$<$ je čiastočné usporiadanie na množine E, nazývané aj vizuálne usporiadanie.*
- *$\mathcal{P}$ je konečná množina instancií.*
- *$\tau \colon E \to \{s, r\}$ je značkovacia funkcia, ktorá rozdelí udalosti na prijímajúce a odosielajúce.*
- *$P \colon E \to \mathcal{P}$ je funkcia ktorá, asociuje každú udalosť s práve jedným procesom.*

- $\mathcal{M} \subseteq (\tau^{-1}(s) \times \tau^{-1}(s))$ *je bijektívne mapovanie, také že každej odosielajúcej udalosti priradí unikátnu prijímajúcu udalosť.*

- $\mathcal{K} \subseteq \{ (e, f, o) \mid e, f \in E, e < f, o \in \mathcal{I}\}$ *je množina relatívnych časových obmedzení.*

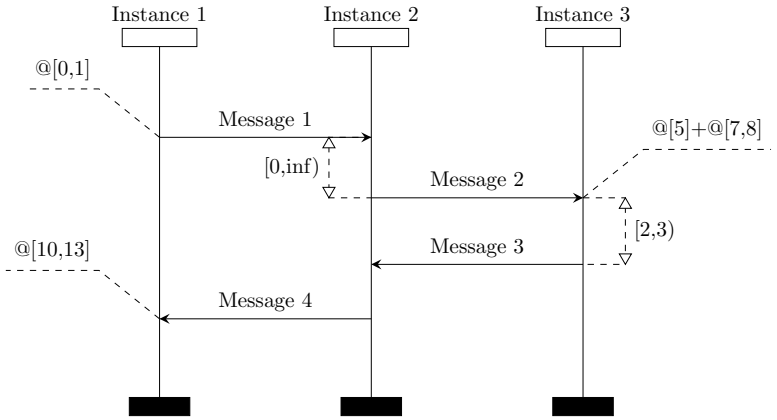
- $\mathcal{L} \subseteq \{ (e, o) \mid e \in E, o \in \mathcal{I}\}$ *je množina absolútnych časových obmedzení.*

Definícia 3. *Vizuálne usporiadanie je potom definované ako reflexívny a tranzitívny uzáver* $\mathcal{M} \cup \bigcup_{p \in \mathcal{P}} <_p \; kde \; <_p = \prec_p \; \wedge \; \prec_p$ *je totálne usporiadanie na* $\mathcal{P}^{-1}(p)$.

Z predchádzajúcich definícii vyplýva, že každú komponentu popisovaného systému reprezentuje jedna instancia. Instancie medzi sebou komunikujú zasielaním asynchrónnych správ. Prijatie aj odoslanie správy sú udalosti a každá udalosť leží na práve jednom procese. Udalosti môžu byť dvoch typov, prijímajúce udalosti a odosielajúce udalosti. Prijímajúce alebo odosielajúce udalosti nastanú na instancii pri prijatí respektíve odoslaní správy. Ďalej sú udalosti na instacii usporiadané v poradí v akom nastali a to sa nemôže za žiadnych okolností meniť. Medzi instanciami sú udalosti usporiadané v smere správy tzn. zasielajúca udalosť je vždy pred prijímajúcou udalosťou. Časové údaje v MSC sú reprezentované intervalmi a môžu byť dvoch typov, absolútne a relatívne. Absolútne časy sa vzťahujú ku globálnemu času, zatiaľ čo relatívne časy sa vzťahujú k dvom vizuálne usporiadaným udalostiam [6].

A teraz si ilustrujeme definíciu MSC na príklade z obrázku 1.1. Grafická reprezentácia BMSC je množina zvislých čiar reprezentujúcich instancie a množina horizontálnych čiar reprezentujúcich správy. Instancie aj správy sú pomenované. Na obrázku sa nachádzajú tri instancie a štyri správy. Odosielajúca udalosť je zobrazená ako bod, z ktorého vychádza šípka a prijímajúca udalosť ako patričná vstupná šípka. Udalosti sa môžu vyskytnúť len na instancii, nikdy nie vedľa instancie. Absolútne časové intervaly sú pripojené k jednej udalosti. Značíme ich symbolom @ pred samotným intervalom, ako môžeme vidieť na prvej udalosti *Instancie 1*. Relatívne časy sú znázornené ako prerušovaná obojsmerná šípka medzi dvomi udalosťami, napríklad medzi prvou a druhou udalosťou na Instacii 3 je časové obmedzenie s hodnotou [2,3). Jedna udalosť môže zároveň obsahovať viacero relatívnych časových obmedzení aj relatívny a absolútny časový interval súčasne. Vždy však môže obsahovať maximálne jeden absolútny časový interval naraz. Ako vidíme na prvej udalosti *Instancie 3*, BMSC môže obsahovať aj zjednotenie časových intervalov, ktoré je reprezentované symbo-

lom + medzi jednotlivými intervalmi. To však uvádzame len ako referenciu a v našej práci sa tým bližšie nebudeme zaoberať.



Obr. 1.1: Elementy MSC

Pre časové obmedzenia v MSC diagramoch platí nasledujúca definícia prevziata z [7].

Definícia 4. *Pre BMSC* $M = (E, <, \mathcal{P}, \tau, \mathcal{P}, \mathcal{M}, \mathcal{K}, \mathcal{L})$ *definujeme časové priradenie* ι *ako funkciu* $\iota : E \to \mathbb{R}^+$ *ktorá priradí každej udalosti časovú známku tak, že platí:*

- $\forall e, f \in E, \quad e < f \implies \iota(e) \leq \iota(f)$
- $\forall (e, f, o) \in \mathcal{K}.\ \iota(f) - \iota(e) \in o$
- $\forall (e, o) \in \mathcal{L}.\ \iota(e) \in o$

Časové priradenie špecifikuje, kedy nastala daná udalosť. BMSC reprezentuje množinu všetkých časových priradení. Takúto množinu budeme volať množina riešení.

Množinu riešení si môžeme predstaviť ako priradenie každej udalosti práve jeden konkrétny čas z množiny jej časových obmedzení. Pre prvú udalosť na *Instanci 1* z obrázku 1.1 zvoľme 1, pre druhú udalosť 10 atď. Nasledujúca definícia je použitá z [7].

Definícia 5. *BMSC je časovo konzistentné, ak množina riešení nie je prázdna.*

1.2 Sequence Chart Studio (SCStudio)

SCStudio je užívateľský prívetivý open-source nástroj na kreslenie a verifikáciu MSC, ktorý je vyvíjaný v Inštitúte teoretickej informatiky na Fakulte informatiky MU v spolupráci s firmou Siemens Convergence Creators, s.r.o [3]. SCStudio obsahuje dve časti. Prvá časť je multiplatformná a obsahuje širokú sadu algoritmov [1]:

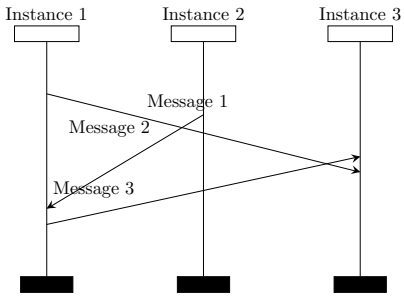
- Import a export z rozličných formátov.
- Algoritmy na verifikáciu, ktoré overia či MSC splňuje zadané vlastnosti. Tieto algoritmy sú implementované ako checker, ktorý v prípade chyby vráti užívateľovi na výstup protipríklad vo forme MSC s vyznačenou chybou.
- Algoritmy, ktoré menia MSC v určitom význame. Tieto algoritmy sú naimplementované ako transformery, ktoré nevrátia nič na výstup, ale zmenia vstupné MSC.

Pre obsluhu týchto algoritmov je vytvorené užívateľsky prívetivé rozhranie pre prácu z príkazovej riadky a rozsiahla dokumentácia. Druhá časť je plugin do Microsoft Office Visio, ktorý pracuje s prvou časťou a zároveň poskytuje grafické rozhranie.

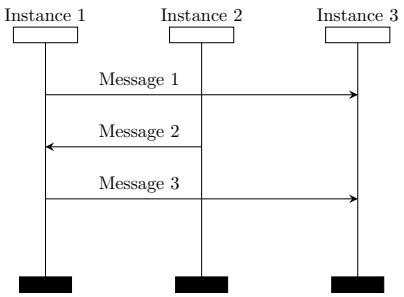
1.2.1 Beautify

Transformer *beautify* poskytuje funkcionalitu, ktorá prekreslí dané MSC do usporiadanejšieho a čitatelnejšieho tvaru. *Beautify* pracuje nad všetkými prvkami podporovanými štandardom a obsahuje rozsiahle užívateľske nastavenia pre optimalizáciu grafického výstupu. Jedným z hlavných využití *beautify* je to, že pri importe z textového formátu Z.120, ktorý neobsahuje všetky grafické informácie, optimálne rozmiestni MSC diagram [9]. Na obrázkoch 1.2, 1.3 môžeme vidieť dva MSC diagramy z rovnakou textovou reprezentáciou, ale druhý diagram je kvôli lepšiemu zobrazeniu prekreslený transformerom *beautify*. Transformer *beautify* obsahuje dva možné spôsoby rozvrhovania udalostí [9]. Prvá možnosť je založená na lineárnom programovaní a nástroji na riešenie lineárnych rovníc *lp_solve*. Nástroj *lp_solve* vyrieši presne všetky podmienky pre užívateľský prívetivé vykreslenie MSC, ale má aj svoje nevýhody. Hlavnou nevýhodou je počet udalostí, ktoré dokáže spracovať, respektíve čas, v ktorom pre dané MSC dokončí výpočet. Implementácia *lp_solve*, ktorú využívame, zvládne spracovať najviac 400 udalostí, čo je napríklad pre import z pcapu značne ob-

medzujúce. Druhá možnosť rozvrhovania udalostí v beautify je založená na heuristike, ktorá nie je obmedzovaná počtom udalostí. Táto heuristika je značne rýchlejšia ako *lp_solve*, ale môže byť nepresná. V nastavení *beautify* je potrebné explicitne určiť, ktorá možnosť sa použije.



Obr. 1.2: MSC pred beautify



Obr. 1.3: MSC prekreslené beautify

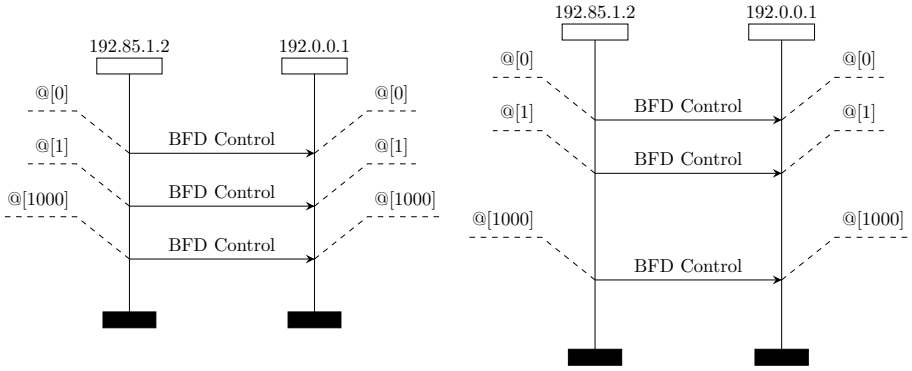
1.3 Motivácia

Stávajúca implementácia SCStudia poskytuje funkcionalitu pre import z formátu pcap. Pcap (packet capture) je formát zachytávajúci sieťovú komunikáciu. V najjednoduchšom nastavení jeden zachytený paket reprezentuje jednu správu. Zdrojová a cieľová IP adresa predstavuje instancie, medzi ktorými bola správa odoslaná. Obsah dát z packetu, ako aj informácie o použitom protokole tvoria správu medzi týmito dvomi instanciami. Čas odoslania, respektíve prijatia paketu je čas prijímajúcej, respektíve odosielajúcej udalosti, ako môžeme vidieť na obrázkoch 1.4 a 1.5 vľavo. Kedže *beautify* neberie do úvahy časové značky, všetky správy sú od seba rovnako vzdialené. Podľa časových značiek je vhodné tretiu správu odsadiť o väčšiu vzdialenosť, lebo nastala v neskoršom čase ako je znázornené na obrázku 1.5 vpravo. Ďalej je možné zobrazované informácie filtrovať podľa vrstvy TCP/IP modelu a zároveň agregovať viacero správ do jednej v závislosti na zvolenej sémantike. Výsledná správa potom bude mať čas odoslania rovný času odoslania prvej agregovanej správy a čas prijatia rovný času prijatia poslednej agregovanej správy. Pri vizualizácii je preto dôležité dodržiavať sémantiku časových značiek a udalosti podľa toho príslušne zobraziť. Chceme dosiahnuť, aby agregovaná správa nebola zobrazená rovno, ale aby bola zobrazená s ohľadom na čas prijatia, respektíve doručenia ako je znázornené na obrázku 1.6 vpravo. Transformer *beautify*

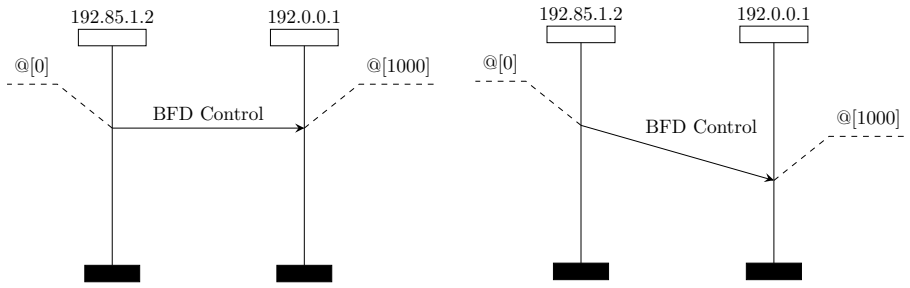
túto funkcionalitu neposkytuje, preto sme vytvorili transformer *time relevant ordering*, ktorý pri zobrazovaní MSC berie časové značky do úvahy. Ďalej je potrebné si všimnúť, že importované súbory obsahujú často veľké množstvo udalostí. Preto je vyžadované, aby bol *time relevant ordering* čo najrýchlejší. Import z formátu pcap používa vždy len jeden typ časových značiek, teda buď len relatívne alebo len absolútne časové značky, čo ďalej špecifikuje zadanie pre *time relevant ordering* [4]. Z uvedených argumentov vyplýva, že transformer *beautify* je vhodný pre vykresľovanie MSC, ale pre vykreslenie pcap súboru je vhodnejší novovytvorený transformer *time relevant ordering*.

Time	Source	Destination	Protocol	Length	Info
0.100	192.85.1.2	192.0.0.1	BFD Control	79	Diag:No Diag
0.200	192.85.1.2	192.0.0.1	BFD Control	79	Diag:No Diag
3.000	192.85.1.2	192.0.0.1	BFD Control	79	Diag:No Diag

Obr. 1.4: Textová reprezentácia pcap súboru zachyteného v programe Wireshark



Obr. 1.5: Vľavo je pcap súbor naimportovaný v MS Visio s použitím transformeru *beautify*. Vpravo je ten istý pcap súbor naimportovaný pomocou transformeru *time relevant ordering*.



Obr. 1.6: Na oboch obrázkoch je použitá agregácia na MSC z 1.5. MSC vľavo je zobrazené pomocou *beautify*, vpravo pomocou transformeru *time relevant ordering*.

2 Analýza

Cieľom tejto kapitoly je zadefinovať formát vstupných dát a požiadavky na funkcionalitu, ako aj popísať vybrané algoritmy, ktoré táto práca rozširuje, alebo na ne nadväzuje a ich problémy. Je potrebné si všimnúť rozdiely medzi novo vytvorenými a súčasnými algoritmami.

2.1 Formát vstupných dát a import z formátu pcap

Pre bližšie zadefinovanie vstupných dát potrebujeme nasledujúcu definíciu, ktorá je prevzatá z [1], [2].

Definícia 6. *BMSC* $B = (E, <, \mathcal{P}, \tau, \mathcal{P}, \mathcal{M}, \mathcal{K}, \mathcal{L})$ *je acyklické, ak jeho vizuálne usporiadanie* $<$ *je antisymetrické, neobsahuje cyklus okrem cyklov dĺžky 1.*

Podmienka acykličnosti zaručuje, že neexistuje cyklus vo vizuálnom usporiadaní na B. Ak táto podmienka neplatí, znamená to že v danom BMSC je správa, ktorá bola doručená skôr ako bola poslaná. Takéto chovanie systému nie je správne a je zároveň neimplementovateľné.

Transformer *time relevant ordering* je primárne určený pre import z formátu pcap, ktorý nám obmedzí množinu vstupov len na prvky, ktoré sa v ňom môžu vyskytnúť. Transformer *time relevant ordering* dostáva na vstup MSC diagram, ale v skutočnosti pracuje len s BMSC a HMSC deleguje na transformer *beautify*. Dôvod, prečo nepracuje aj s HMSC je ten, že pre HMSC zobrazenie podľa časov nemá zmysel a výsledok by bol rovnaký ako pri beautify. Predpokladom pre správne fungovanie algoritmu *time relevant ordering* je acyklickosť vstupného BMSC, ktorú import z formátu pcap dodržuje. Ďalej vstup nemôže obsahovať koregión, oblasť v ktorej udalosti nie sú medzi sebou usporiadané a vždy bude obsahovať len jeden typ časových značiek, buď absolútne alebo relatívne. Časové značky nemusia byť zadané úplne. Pre absolútne časy to znamená, že nie všetky udalosti obsahujú časovú značku. V rámci relatívnych časov predpokladáme pre všetky vizuálne usporiadané udalosti polouzavretý interval [0,∞), ktorý vyplýva z vlastnosti vizuálneho uspriadania. Interval [0,∞) znamená, že vizuálne vyššia udalosť nastane v intervale [0,∞) po vizuálne nižšej udalosti. Algoritmus d'alej pracuje len s jednoduchými časovými intervalmi a zjednotenia časových intervalov ignoruje. Import z formátu pcap nemá žiadne obmedzenia na počet udalostí. Je potrebné ale brať ohľad na vykresľovacie možnosti Microsoft Visio – cca 2000 udalostí [9]. V prípade príkazovej riadky a exportu do LaTeXu nie sú kladené žiadne obmedzenia.

2.2 Algoritmy zjemňovania a konzistentnosti časových intervalov

Najskôr si zadefinujeme zjemňovanie a potom si obrázku vysvetlíme jeho fungovanie. Nasledujúce definície sú použité z [5], [7].

Definícia 7. *Pre intervaly $o, r \in \mathcal{I}$, definujeme čiastočné usporiadanie $\preceq \subseteq \mathcal{I} \times \mathcal{I}$ také že $o \preceq r$ platí práve vtedy ak platí:*

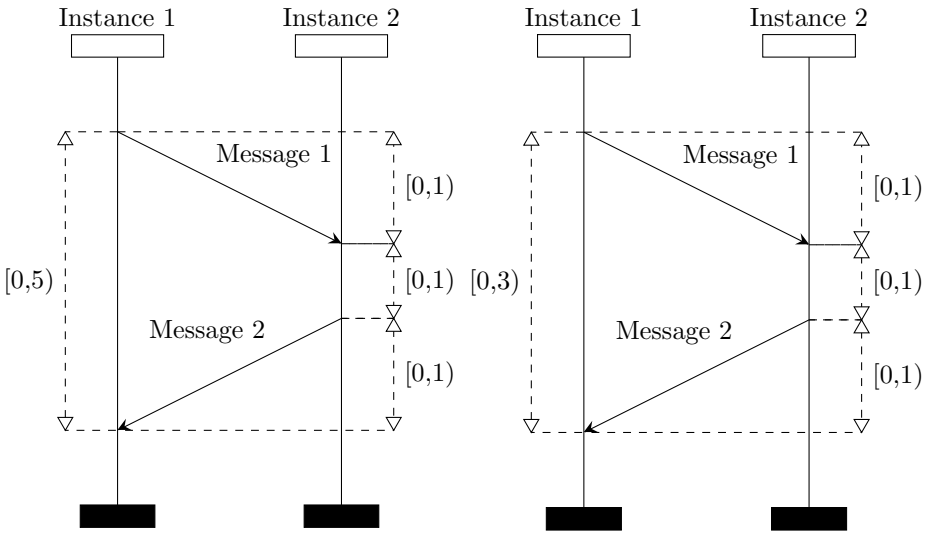
$$\forall\, x.\ x \in o \Rightarrow x \in r.$$

Majme BMSC M a N, ktoré sa líšia len hodnotami intervalov. Potom A je jemnejšie ako B, $A \preceq B$ práve vtedy ak pre každý interval o z A existuje príslušný interval d z B taký že $o \preceq r$. Dve BMSC sú ekvivalentné ak sa líšia len hodnotami intervalov a reprezentujú rovnakú množinu riešení.

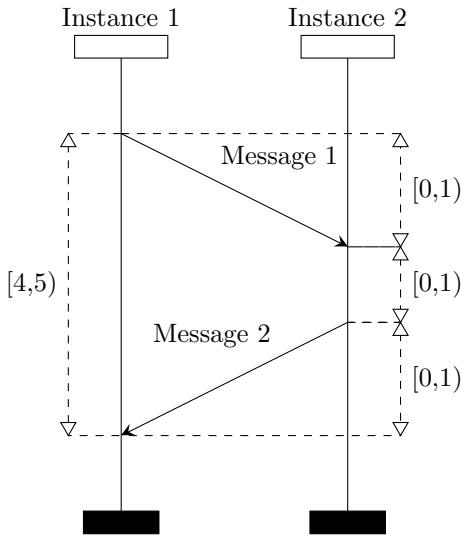
Definícia 8. *Definujme zjemnené BMSC ako minimálne BMSC zo všetkých ekvivalentných BMSC podľa $\preceq$. Problém nájdenia zjemneného BMSC sa nazýva zjemňovanie.*

Všimnime si, že takéto zjemnené BMSC je len jedno. Algoritmy zjemňovania a časovej konzistentnosti majú veľa spoločného a pracujú na veľmi podobnom princípe. Algoritmus zjemňovania je v SCStudiu realizovaný ako transformer *tighter* a algoritmus časovej konzistentnosti ako *time consistency checker*. Transformer *tighter* [8] pracuje nad časovo konzistentným MSC a zjemní množinu všetkých časových intervalov tak, že výsledná množina je ekvivalentná s množinou časových priradení nad daným MSC, ako môžeme vidieť na obrázku 2.1. Z obrázku vidíme, že interval [0,5) nie je celý splniteľný, lebo intervaly [0,1) z druhej strany ho obmedzujú len na interval [0,3). Časové priradenie ι preto nikdy nenadobudne hodnoty z intervalu [3,5). A tak transformer *tighter* zjemní interval [0,5) na interval [0,3). *Time consistency checker* skontroluje časovú konzistenstnosť daného MSC. Na obrázku 2.2 je príklad nekonzistentného MSC. Interval [4,5) je v konflikte s intervalmi [0,1) na druhej strane a preto je množina riešení tohto MSC prázdna. *Time consistency checker* vráti na výstup zadané MSC a vyznačí interval [4,5) ako chybu v konzistentnosti. Problém zjemňovania je ekvivalentný problému časovej konzistentnosti a oba problémy sa riešia rovnako. MSC je časovo nekonzistentné, ak algoritmus zjemňovania nad daným MSC vráti aspoň pre jednu udalosť prázdny interval. Terajšia implementácia týchto algoritmov pracuje len s relatívnymi časovými značkami a absolútne časové značky sú ignorované. Tieto algoritmy narozdiel

od transformeru *time relevant ordering*, pracujú aj so zjednoteniami časových intervalov, čo značne zvyšuje ich zložitosť až na exponencionálnu. V prípade len jednoduchých intervalov je použitý algoritmus založený na Floyd–Warshalle so zložitosťou $\mathcal{O}\left(n^3\right)$ [7].



Obr. 2.1: Vľavo je MSC s nezjemnenými intervalmi. Vpravo to isté MSC po zjemnení intervalov.



Obr. 2.2: Nekonzistentné MSC

3 Rozmiestňovanie

V tejto kapitole popisujeme možné spôsoby rozmiestnenia a implentovaný algoritmus rozmiestňovania udalostí podľa časových značiek. Priblížime si jeho zložitosť, spôsob fungovania a zadefinujeme si užívateľské nastavenia.

3.1 Problém časových intervalov

Pri rozvrhovaní podľa časových intervalov sa ponúka otázka, ktorý časový údaj $(e, o) \in \mathcal{L}$ je najrelevantnejší, najintuitívnejší a bude najlepšie spĺňať podmienky zadania. Z podstaty časového intervalu nám vyplýva, že časová známka t, podľa ktorej bude udalosť zobrazená leží v intervale $o = (a, b)$. Rozvrhovaním udalostí sa snažíme priradiť všetkým udalostiam len y súradnicu, lebo udalosť musí ležať na jej prislúchajúcej instancii. Rozvrhovaním instancií sa zaoberáme v sekcii 3.3. Nemôžeme použiť ľubovoľnú hodnotu z intervalu o, lebo interval o môžeme predtým zjemniť. Zjemnený interval $o' = (a', b')$ nám dáva na výber niekoľko možností ako nájsť túto časovú známku t:

1. $t = a'$
2. $t = b'$
3. $t = (a' + b')/2$
4. $t = f(a', b')$, kde f je ľubovoľná iná funkcia

Testovaním v rámci projektu SCStudia sme prišli k názoru, že najlepšia bude prvá možnosť, nakoľko je najintuitívnejšia a hodí sa pre všetky typy diagramov, narozdiel od druhej možnosti, ktorej výsledky záviseli od konkrétneho diagramu. Ďalej môže byť časový interval o' typu uzavretý, otvorený alebo polootvorený. Táto informácia by nemala byť braná v úvahu, takže intervaly [a,b), [a,b] a (a,b) budú zobrazené na rovnakej úrovni ak sa nevyskytnú žiadne ďalšie obmedzenia. Túto časovú známku t ale nemôžeme jednoducho priradiť ako y súradnicu, lebo viacero udalostí na rovnakej instancii môže mať túto časovú známku rovnakú a boli by zobrazené na rovnaké miesto. Preto bolo potrebné použiť iný spôsob ako je ukázané v nasledujúcich podkapitolách.

3.2 Predspracovanie MSC

Algoritmus začína jednoduchým prechodom celého MSC, s úlohou zistiť časové značky nachádzajúce sa v ňom. Na základe zistených časov v prípade výskytu oboch typov časových značiek deleguje ďalšie rozmiestnenie na transformer *beutify*. Táto situácia by nemala nikdy nastať, lebo import z formátu pcap používa vždy iba jeden typ časových značiek. V opačnom prípade zjemní MSC podľa typu značiek, ktoré obsahuje. Ak MSC obsahuje relatívne časové značky, použije sa terajší algoritmus, inak nová verzia algoritmu popísaná v ďalšej kapitole. Následne algoritmom 3.1 zoradíme udalosti do zoznamu podľa časových značiek. Algoritmus prechádza v cykle prvú udalosť každej instancie s cieľom nájsť najvhodnejšieho kandidáta na umiestnenie na základe časovej značky. Keď ju nájde, vloží ju do zotriedeného zoznamu udalostí a otestuje jej párovú udalosť na zhodu časových značiek, kvôli snahe o rovné vykreslenie správ. Po spracovaní všetkých udalostí algoritmus končí a vracia zotriedený zoznam, ktorý si vybudoval. Tento algoritmus má zložitosť $\mathcal{O}\left(\,|n| \times |m|\,\right)$: n je počet instancií, m je maximum z počtu udalostí na každej instancii. Výsledkom predspracovania je zotriedený zoznam udalostí v poradí, v akom budú rozmiestené.

3.3 Rozmiestňovanie udalostí v MSC

V tejto časti približujeme konkrétny spôsob rozmiestňovania prvkov v MSC. Niektoré algoritmy sú ponechané z implementácie beautify a preto budú popísané len stručne. Prvý algoritmus je `InstanceSequencer` z modulu beautify. Zaoberá sa rozmiestnením a usporiadaním instancií a má zložitosť $\mathcal{O}\,(n!)$ vzhľadom k počtu instancií [9]. Po jeho skončení sú všetky instancie umiestnené a to sa už nebude meniť. Vzhľadom na relatívne malý počet instancií v MSC je tento algoritmus postačujúci a preto nebolo potrebné implementovať jeho novú variantu.

Vzdialenosti dvoch susedných udalostí z usporiadaného zoznamu môžu byť konštantné alebo logaritmicky závislé na rozdiele hodnôt časov týchto udalostí. Rozmiestnením y súradníc podľa tohto kritéria sa zaoberajú algoritmy `ConstantValueOrder` a `LogarithmicValueOrder`. Je dôležité si všimnúť, kedže v tomto štádiu sú instancie už rozmiestnené a y súradnice udalostí pridelené jedným z vyššie zmienených rozmiestňovacích algoritmov, že je potrebné už len rozmiestnenie časových značiek a rozmiestňovanie bude dokončené. O to sa stará `TimeTransformer` z beautify [9].

Algorithm 3.1: Usporiadanie udalostí podľa časových značiek

Vstup : BMSC $bmsc$
Výstup: zotriedený zoznam udalostí

```
1  List⟨Event⟩ sorted_events
2  List⟨Instance⟩ instances ← bmsc.get_instances()
                    // iterate until all events on all instances will be processed
3  while not instances.empty() do
4      Event first_event = NULL
                    // each iteration finds best fit event
5      foreach instance ∈ instances do
6          if instance.get_first_event() == NULL then
7              instances.remove(instance)
8              continue
9          Event front_event = instance.get_first_event()
10         if first_event == NULL then
11             first_event = front_event
12             continue
13         if get_time(front_event) < get_time(first_event) then
14             first_event = front_event
15     if first_event ≠ NULL then
16         sorted_events.insert(first_event)
17         instances.remove(first_event.get_instance())
18         Event matching_event ← first_event.get_matching_event()
19         Instance matching_event_ins = matching_event.get_instance()
20         if matching_event == matching_event_ins.get_first_event() &&
           get_time(matching_event) == get_time(first_event) then
21             sorted_events.insert(matching_event)
22             matching_event_instance.remove(matching_event)
```

3.3.1 Nastavenia rozmiestňovania

Štruktúra pre nastavenia obsahuje tieto položky:

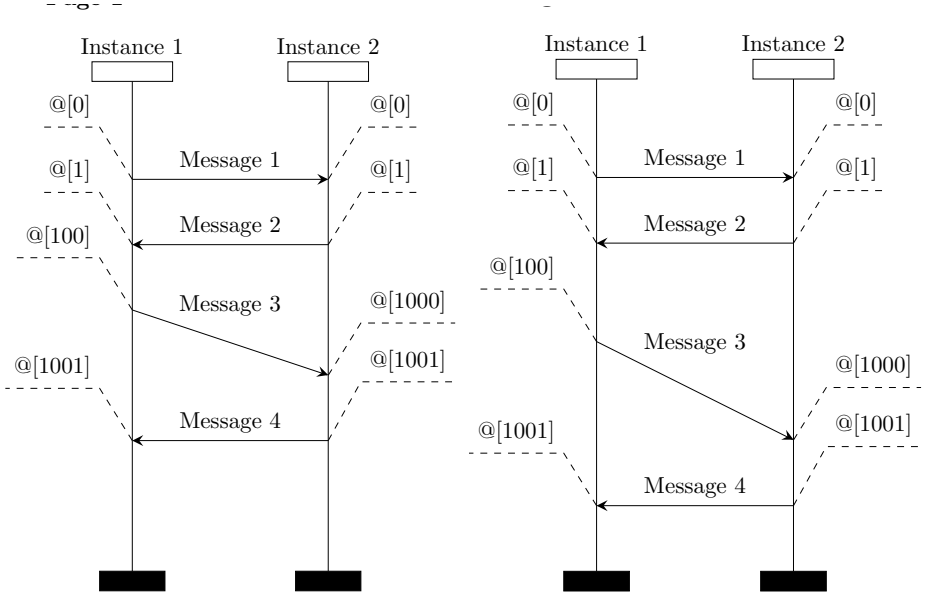
- `min_shift` – minimálna vzdialenosť medzi dvoma susednými udalosťami. V prípade konštantného rozmiestňovania je to hodnota, ktorá sa použije ako vzdialenosť medzi dvoma udalosťami, v prípade logaritmického rozmiestňovania je to minimálna vzdialenosť medzi dvoma instanciami.
- `max_shift` – maximálna vzdialenosť medzi dvoma susednými udalosťami. Používa sa len v prípade logaritmickeho rozmiestňovania.
- `is_const_order` – hodnota, ktorá určuje ako bude výsledne MSC rozmiestnené. Hodnota 0 znamená konštantné a hodnota 1 logaritmické vzdialenosti.

Tieto nastavenia sú implicitne nastavené na hodnoty:
$min\_shift = 8$, $max\_shift = 16$ a $is\_const\_order = 1$
pre užívateľsky najprívetivejšie výstupy.

3.3.2 Algoritmy rozmiestňovania

Základ algoritmov je lineárny prechod zotriedeného zoznamu, ktorý sme získali predspracovaním. Algoritmus si udržiava ukazovateľ na aktuálnu úroveň, vzdialenosť od začiatku. Ak má prechádzaná udalosť rovnakú časovú značku t ako predchádzajúca udalosť a leží na rôznej instancii, spadá do aktuálnej úrovne. V tom prípade je jej priradená y súradnica tejto úrovne. Inak sa zvýši hodnota úrovne a udalosti sa priradí y súradnica novej úrovne. Algoritmy rozmiestňovania sa líšia len spôsobom, akým zvyšujú pozíciu tejto úrovne. Pri konštantnom rozmiestňovaní sa pozícia úrovne zvýši o hodnotu min_shift z nastavení, pri logaritmickom rozmiestňovaní sa pozícia úrovne zvýši o $min\_shift \times log(diff)$, kde diff je rozdiel časových značiek aktuálnej a predchádzajúcej udalosti. Tieto algoritmy sú prezentované na obrázku 3.1. Diagram vľavo je rozmiestnený konštantným rozmiestňovaním. Prvá udalosť na druhej instancii spĺňa podmienky, a preto je jej pridelená pozícia rovnakej úrovne ako prvej udalosti na prvej instancii. Udalosť s absolútnym časom 100 je od predchádzajúcich udalostí vzdialená o konštantu min_shift, jej párová udalosť s absolútnym časom 1000 už ale nespadá do rovnakej úrovne, a preto sa pozícia úrovne zvýši. Diagram vpravo je vykreslený logaritmickým rozmiestňovaním. Udalosť s absolútnym časom 100 je od predchádzajúcich udalostí

vzdialená o $min\_shift \times log(99)$, jej párová udalosť s absolútnym časom 1000 je od nej vzdialená o $min\_shift \times log(900)$. Posledná správa je potom od udalosti z časom 1000 vzdialená $min\_shift \times log(1)$. Všetky tieto hodnoty vzdialeností avšak musia spadať do rozsahu $[min\_shift, max\_shift]$.



Obr. 3.1: Vľavo je prekreslené MSC konštantným rozmiestnením, vpravo logaritmickým.

4 Rozširujúce algoritmy

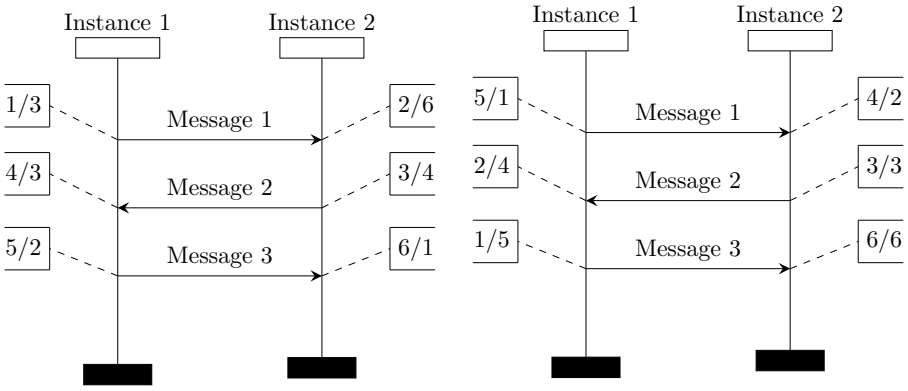
Ako bolo spomenuté v kapitole 2 časové informácie obsiahnuté v MSC diagrame môžu byť značne nepresné. Kedže cieľom algoritmu *time relevant ordering* je zobraziť MSC čo najreálnejšie, musíme pre time relevant ordering zaviesť predpoklad časovej konzistentnosti vstupného MSC diagramu. Inak by boli zobrazené diagramy značne kompromitované. Stávajúca implementácia *time consistency checker* prácu s absolútnymi časmi neumožňovala, preto bolo potrebné ju doimplementovať. Ďalším predpokladom pre kvalitnejšiu prácu s absolútnymi intervalmi bolo pridanie chýbajúcej podpory pre zjemňovanie absolútnych časových značiek. Pre správne fungovanie týchto algoritmov chýbala v SCStudiu podpora prechodu grafom. Bolo preto nutné takúto funkcionalitu k SCStudiu pridať. Táto kapitola preto prezentuje všetky potrebné doplňujúce algoritmy, ktoré boli začlenené do SCStudia.

4.1 Prechod MSC diagramom

Prechod MSC diagramom je v SCStudiu realizovaný pomocou traverserov a listenerov. Traversery využívajú k prechodu grafom algoritmus depth first search (DFS), ktorý si značkuje uzly na základe farieb. Nenavštívené uzly sú biele, navštívené, ale neuzavreté uzly sú sivé a uzavreté uzly sú čierne. Traversery majú jednotné rozhranie a pre každý typ, farbu nájdeného uzlu, sú schopné zaregistrovať listener. Tento listener obsahuje funkcionalitu, ktorá sa vykoná pri nájdení uzlu danej farby. V časovo relevantnom usporiadaní sa na MSC diagrame využívajú predovšetkým traversery `DFSEventsTraverser` a `DFSInstanceEventsTraverser`. Prvý menovaný prechádza MSC do hĺbky preferujúc prijímajúcu udalosť pred udalosťou nasledujúcou na instancii, zatiaľ čo `DFSInstanceEventsTraverser` spracováva len nasledujúce udalosti na instancii.

4.1.1 DFSBackwardEventsTraverser

Algoritmy časovej konzistentnosti a zjemňovania vo svojej implementácii pre absolútne časy vyžadujú spracovanie všetkých udalostí nad sebou, to znamená prechod MSC diagramom zdola nahor. Táto podmienka vyplýva z princípu fungovania DFS, ktorý pracuje tak, že keď prefarbuje udalosť načierno už má spracované všetky udalosti dosiahnuteľné z tohto uzlu, tzn. všetky udalosti ktoré vizuálne usporiadané pred touto

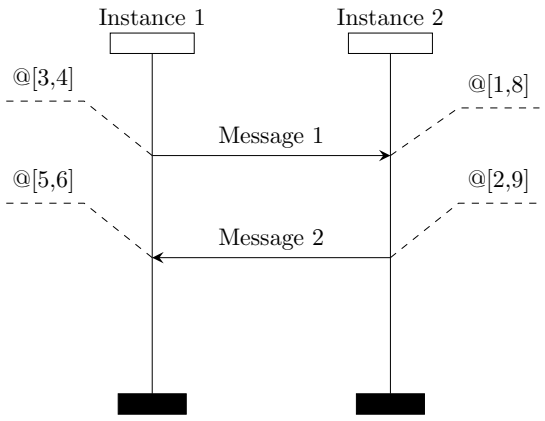


Obr. 4.1: Vľavo je prechod MSC diagramu DFSEventTraverserom, vpravo BackwardTraverserom.

udalosťou. V doposiaľ naimplementovaných traverseroch prechod zdola nahor nebol možný, preto sme zadaný traverser doimplementoval dodržiavajúc interface a začlenili sme ho do štruktúry. Je potrebné si všimnúť rozdiel v prechádzaní párových udalostí medzi `DFSEventsTraverser` a `DFSBackwardTraverser`. `DFSEventTraverser` navštevuje párové udalosti z rovnakej správy len ak daná udalosť je odosielajúca s cieľom dostať sa v grafe k udalostiam, ktoré sú vo vizuálnom usporiadaní neskôr. To vyplýva z vlastnosti MSC, že odosielajúca udalosť je vizuálne usporiadaná pred prijímajúcou udalosťou. Naproti tomu `DFSBackwardTraverser` navštevuje párovu udalosť len ak daná udalosť je prijímajuca s cieľom dostať sa v grafe k udalostiam, ktoré sú vo vizuálnom usporiadaní skôr. Na obrázku 4.1 obsahujú oba MSC diagramy pri každej udalosti komentár typu x/y, kde x je poradie, v ktorom bola udalosť traverserom navštívená a y je poradie, v ktorom bola udalosť spracovaná a prefarbená načierno. Na obrázku vľavo prejde `DFSEventsTraverser` naraz celé MSC. Na obrázku vpravo vidno ako `BackwardTraverser`, pri prechádzaní nenavštívi párovú udalosť, lebo udalosť z označením $1/5$ je odosielajúca, ale ide na svojho predchodcu.

4.2 Zjemňovanie absolútnych intervalov

Ako môžme vidieť na obrázku 4.2, obe udalosti na *Instancii 1* majú pridelené časové intervaly, ktoré ale nie sú celé splniteľné. Prvá udalosť na *Ins-*

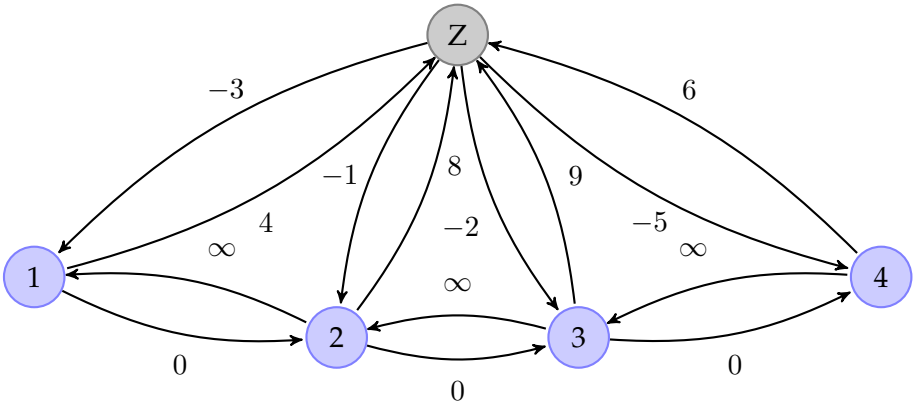


Obr. 4.2: Nezjemnené MSC

tancii 2, nemôže nastať skôr ako jej odosielajúca udalosť. Tým obmedzíme jej dolnú hranicu časového intervalu na [3,8]. Analogicky obmedzíme aj druhú udalosť na *Instancii 2* na interval [3,9], pretože nemôže nastať skôr ako jej predchodca. Posledná udalosť na *Instancii 1* môže nastať najneskôr v čase 6, avšak jej odosielajúca udalosť až v čase 9. Preto obmedzíme hornú hranicu jej odosielajúcej udalosti na [3,6]. Aby bol graf zjemnený úplne musíme ešte zjemniť hornú hranicu prvej udalosti na *Instancii 2* na interval [3,6]. Kvôli absencii podpory pre absolútne časové intervaly v už naimplementovanom algoritme zjemňovania bolo potrebné doimplementovať verziu pre absolútne časové intervaly.

4.2.1 Problém zjemňovania absolútnych časových intervalov

Zjemňovanie je obecnejší problém ako časová konzistentnosť, preto popisujeme iba zjemňovanie. Princíp zjemňovania je založený na riešení problému jednoduchých časov, (*Simple temporal problem*, STP) [5]. STP je dobre známy problém. STP je definovaný na orientovanom grafe s intervalmi na hranách. Časová konzistentnosť a zjemňovanie je pre STP zadefinované podobne ako pre BMSC. My pracujeme s MSC, ktoré ale vieme jednoducho previesť na STP. Vyrobíme si nový minimálny uzol Z. K nemu natiahneme pre každú udalosť z MSC neorientovanú hranu z hodnotou jej absolútneho časového intervalu. Ďalej pre každú dvojicu vizuálne usporiadaných udalostí vyrobíme orientovanú hranu s hodnotou $[0,\infty)$ z vizu-



Obr. 4.3: MSC diagram prevedený na graf z váženými cestami

álne nižšej udalosti do vizuálne vyššej. To nám vynúti vizuálne usporiadanie [8]. Vyriešiť zjemňovanie na STP znamená nájsť hodnoty najkratších ciest na orientovanom grafe s váženými hranami. Nájdenie najkratšej cesty z každého vrcholu do každého vrcholu na grafe závislotí vyrieši STP. MSC s absolútnymi časmi z predchádzajúceho obrázku 4.2 vyrobíme na graf závislostí $G = (V, H)$ nasledujúcim spôsobom. Korektnosť tohoto postupu je dokázaná v [5], takže týmto sa ďalej nebudeme zaoberať.

Definícia 9. *Graf závislostí $G = (V, H)$ pre MSC $M = (E, <, \mathcal{P}, \tau, \mathcal{P}, \mathcal{M}, \mathcal{K}, \mathcal{L})$ definujeme nasledovne:*

- *Vrcholy grafu sú $V = E \cup \{Z\}$, kde Z je nový počiatočný vrchol.*
- *Hrany grafu sú $H \subseteq V \times V \times (\mathbb{R} \cup \{-\infty, \infty\})$, kde každá hrana je jedným z nasledujúcich typov.*
  1. (Z, e, w), *ak* $\exists\, (e, o) \in \mathcal{L}$ *taká, že* $-w$ *je dolná hranica intervalu* o.
  2. (e, Z, w), *ak*$\exists\, (e, o) \in \mathcal{L}$ *taká, že* w *je horná hranica intervalu* o.
  3. $(e, f, 0)$, *ak* $\exists e, f \in E.\ e < f$. *Všetkým vizuálne nižším udalostiam pridáme hranu s váhou 0 do všetkých vizuálne vyšších vrcholov.*
  4. (e, f, ∞), *ak* $\exists e, f \in E.\ e > f$. *Všetkým vizuálne vyšším udalostiam pridáme hranu s váhou* ∞ *do všetkých vizuálne nižších vrcholov.*

Výsledný graf závislostí pre obrázok 4.2 je znázornený na obrázku 4.3. Kvôli prehľadnosti sú vynechané tranzitívne hrany s váhami 0 a ∞.

Pre hľadanie najkratších ciest sa dá použiť algoritmus Floyd–Warshall, v tomto prípade je ale efektívnejšie spojenie algoritmov 4.1, 4.2, ktoré sme

vymysleli a ich korektnosť pre BMSC vygenerované pcap filtrom dokazujeme nižšie.

Veta 1. *Nech MSC M je časovo konzistentné, potom spustenie algoritmov 4.1, 4.2 na M a zkonštruovanie grafu závislostí vráti rovnaký hrany do vrcholu Z ako spustenie algoritmu Floyd–Warshall na grafe závislostí M.*

Dôkaz 1. *Prezentujeme iba hlavnú myšlienku dôkazu. Pre najkratšiu cestu c z vrcholu Z do vrcholu e platí:*

- *Z predpokladu časovej konzistentnosti vyplýva, že graf závislostí neobsahuje cykly s negatívnou váhou. Potom cesta c nemôže obsahovať cyklus cez vrchol Z, lebo platí podmienka, že pre každý interval z dolnou hranicou a a hornou hranicou b, $a \leq b$. Preto váha cyklu $Z \cdot e \cdot Z, e \in E$ nebude nikdy menšia ako nula, lebo návrat do vrcholu Z vždy zdvihne váhu cesty o hodnotu $b - a$.*

- *Cesta c nemôže prechádzať cez vrchol f, kde platí $f > x$, lebo vzostupné hrany majú váhu ∞. Preto cesta cez takéto vrcholy f do vrcholu e vždy zhorší pôvodnú cestu.*

Z toho vyplýva, že najkratšia cesta z vrcholu Z do vrcholu e neobsahuje cykly a môže prechádzať len vrcholmi f_i pre ktoré platí: ak $f_i < ei < e => f_i > f_e$. Preto algoritmus 4.1 hľadá iba také najkratšie cesty, ktoré vedú cez vrcholy $f_i < e$. Zjavne pri rozumnom prechádzaní MSC je nutné pamätať si v každom uzle f_i iba minimálnu váhu cesty z vrcholu Z do f_i. Z využitím tejto informácie z priamych predchodcov e, podľa vizuálneho usporiadania, vypočítame váhu najkratšej cesty zo Z do e.

Pre najkratšiu cestu z vrcholu e do vrcholu Z platí:

- *Z predpokladu časovej konzistentnosti vyplýva, že graf závislostí neobsahuje cykly s negatívnou váhou. Potom cesta c nemôže obsahovať cyklus cez vrchol Z, lebo platí podmienka, že pre každý interval z dolnou hranicou a a hornou hranicou b, $a \leq b$. Preto váha cyklu $Z \cdot e \cdot Z, e \in E$ nebude nikdy menšia ako nula, lebo návrat do vrcholu Z vždy zdvihne váhu cesty o hodnotu $b - a$.*

- *Cesta c nemôže prechádzať cez vrchol f, kde platí $f < e$ lebo vzostupné hrany majú váhu $w_{fe} = \infty$. Preto cesta cez takéto vrcholy f do vrcholu e vždy zhorší pôvodnú cestu.*

Z toho vyplýva, že najkratšia cesta z vrcholu e do vrcholu Z neobsahuje cykly a môže prechádzať len vrcholmi f_i pre ktoré platí $f_i > e$. Preto algoritmus 4.2

hľadá iba také najkratšie cestym, ktoré vedú cez vrcholy $f_i > e$. Podobne ako pri hľadaní najkratšej cesty z vrcholu Z do vrcholu e, pri prechádzaní MSC stačí ak si budeme pamätať v každom uzle f_i iba minimálnu váhu cesty z vrcholu f_i do Z. Potom z využitím informácií z priamych predchodcov f_i vrcholu e z vizuálneho usporiadania, vypočítame váhu najkratšej cesty z e do Z.

Tieto a vyššie uvedené vlastnosti presne spĺňuje algoritmus prechodu grafom do hĺbky a preto môžme problém hľadania najkratších ciest v tomto grafe zjednodušiť na prechod grafom do hĺbky algoritmami 4.1, 4.2.

4.2.2 Zjemňovanie dolnej hranice intervalov a doplnenie chýbajúcich časov

Pre zjemnenie dolnej hranice intervalov potrebujeme postupne prechádzať MSC diagramom zdola od posledných udalostí na instanciách k tým prvým, a teda hľadáme najkratšiu cestu, ktorá vedie cez vrcholy, ktoré sú vizuálne vyššie ako súčasne spracovávaný uzol. V tomto algoritme využívame vlastnosť DFS, že všetky čierne udalosti, sú už upravené a majú najlepší možný čas. Postupne prechádzame MSC až kým nenájdeme uzol, ktorý nemá následníkov ani párovú udalosť, tzn. je vizuálne najnižšie. Najkratšia cesta z vrcholu Z do tohto vrcholu je preto záporná hodnota dolnej hranice jeho intervalu. Zafarbíme ho načierno a pri backtrackingu na základe jeho najkratšej cesty upravujeme udalosti, ktoré sme navštívili na ceste k nemu. Je potrebné si uvedomiť, že pri zafarbovaní uzlu načierno už máme začiernené všetky uzly, ktoré sú vizuálne vyššie ako tento uzol a mám od nich informáciu o ich najkratšej ceste, ktorú môžme využit pri vypočítaní najkratšej cesty do tohto uzlu. Použijeme `DFSBackwardTraverser` a pri ukončovaní udalosti upravíme interval algoritmom 4.1. MSC z obrázka 4.2 začneme prehľadávať na poslednej udalosti prvej instancie. Postupne navštevujeme druhú a prvú udalosť druhej instancie a prejdeme až na prvú udalosť prvej instancie. Táto udalosť nemá ani predchodcu ani nespracovanú párovú udalosť, preto ju uzavrieme. Jej najkratšia cesta z vrcholu Z je v našom prípade -3. Vraciame sa na udalosť, ktorú sme navštívili predtým, uzatvárame a upravíme hodnotu jej najlepšej cesty na -3. Pokračujeme v backtrackovaní a upravíme aj najkratšiu cestu nasledujúcej udalosti. Počiatočná udalosť má najkratšiu cestu -5 a túto hodnotu už zlepšiť nevieme.

Tento algoritmus je tiež vhodný na doplnenie chýbajúcich časových intervalov. Funguje na princípe dedenia najnižšieho možného času. Udalosti bez časovej známky je nastavený čas na hodnotu `black_time`. V prípade,

Algorithm 4.1: Zjemnenie dolných hraníc intervalov

Vstup : Udalosť e
Výstup: Udalosť e so zjemnenou dolnou hranicou absolútneho intervalu

```
1  double predecessor_time = 0
2  double matching_event_time = 0
3  double black_time = 0
4  if e.has_predecessor_events() then
5     predecessor_time ←
      get_time(e.get_predecessor_event())
6  if e.is_receive() then
7     matching_event_time ←
      get_time(e.get_matching_event())
8  if predecessor_time > matching_event_time then
9     black_time ← predecessor_time
10 else
11    black_time ← matching_event_time
12 double current_time = get_time(e)
13 if current_time < black_time then
14    set_time(e, black_time)
```

že udalosť nemá predchodcu a ani párovú udalosť, je jej analogicky nastavený čas 0. Dedenie časov ale pracuje len v rámci metadát pre lepšie vykreslenie a nikdy nemení pôvodné MSC. Zjemňovanie dolnej hranice, algoritmus 4.1, sa využíva v transformery *absolute time tighter* a je aj defaultné aplikované na MSC diagram pred rozmiestnením algoritmom *time relevant ordering*. V druhom prípade platí, že všetky časové intervaly sú zjemnené len ako metadáta, takže v SCStudiu zostanú pôvodné intervaly nezmenené. Práca nad týmito metadátami má zmysel, lebo užívateľ si chce MSC diagram len zobraziť nie ho meniť a metadáta nám poskytnú presnejšie údaje ako zobraziť dané MSC. Výsledok zjemňovania algoritmami 4.1, 4.2 môžme vidieť na obrázku 4.4.

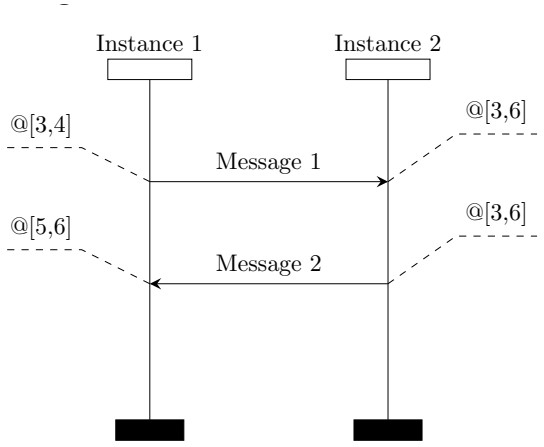
4.2.3 Zjemňovanie hornej hranice intervalov

Tento algoritmus funguje veľmi podobne ako zjemňovanie dolnej hranice. Základný rozdiel je v prechode MSC diagramom. Zatiaľ čo algoritmus zjemňovania dolnej hranice prechádza diagram zdola a pri uzatváraní udalostí im zvyšuje dolnú hranicu, ekvivalentné znižovaniu zápornej hodnoty dolnej hranice, algoritmus zjemňovania hornej hranice prechádza graf zhora a pri uzatváraní udalostí im znižuje hornú hranicu intervalu. MSC z obrázka 4.2 začneme prehľadávať `DFSEventsTraverserom` na prvej udalosti prvej instancie. Postupne navštevujeme prvú a druhú udalosť druhej instancie a prejdeme až na poslednú udalosť prvej instancie. Táto udalosť nemá ani predchodcu ani nespracovanú párovú udalosť, preto ju uzavrieme. Jej najkratšia cesta do vrcholu Z je našom prípade 6. Vraciame sa na udalosť, ktorú sme navštívili predtým, uzatvárame a upravíme hodnotu jej najlepšej cesty na 6. Pokračujeme v backtrackovaní a upravíme aj najkratšiu cestu nasledujúcej udalosti. Počiatočná udalosť má najkratšiu cestu 4 a túto hodnotu už zlepšiť nevieme. Výsledok zjemňovania algoritmami 4.1, 4.2 môžme vidieť na obrázku 4.4.

Algorithm 4.2: Zjemnenie horných hraníc intervalov

Vstup : Udalosť e
Výstup: Udalosť e so zjemnenou hornou hranicou absolútneho intervalu

```
1  double successor_time = ∞
2  double matching_event_time = ∞
3  double black_time = ∞
4  if e.has_successor_events() then
5     successor_time ← get_time(e.get_successor_event())
6  if e.is_send() then
7     matching_event_time ←
      get_time(e.get_matching_event())
8  if successor_time < matching_event_time then
9     black_time ← successor_time
10 else
11    black_time ← matching_event_time
12 double current_time = get_time(e)
13 if current_time > black_absolute_time then
14    set_time(e, black_time)
```



Obr. 4.4: Zjemnené MSC.

Záver

Našou úlohou bolo naštudovať problematiku časov v nástroji SCStudio a doimplementovať doňho variantu grafického rozmiestňovania MSC, ktoré bude brať do úvahy časové značky. Na začiatku sme si predstavili formalizmus MSC, nástroj SCStudio a algoritmy, ktoré poskytuje pre prácu s časovými značkami.

Potom sme si zadefinovali import z formátu pcap ako zatiaľ jedinú možnú aplikáciu novovytvoreného transformeru time relevant ordering a zanalyzovali formát vstupných dát poskytovaný týmto importom. Neskôr sme bližšie preskúmali algoritmy z SCStudia, či už transformer beautify alebo algoritmy na prácu s časom. To nám dalo detailnejšiu predstavu o potrebách SCStudia.

Pri implementácií sme nenarazili na väčšie problémy. Priebežná implementácia bola vždy zkonzultovaná v rámci projektu SCStudio, aby výsledný program čo najlepšie spĺňal dané potreby. Program je samozrejme doplnený o automatickú sadu testov, čo bolo aj súčastou zadania. Testy fungujú tak, že pre každý testovací súbor, je vopred vygenerovaná množina správnych výsledkov. V prípade, že výstup aktuálnej funkcionality sa nenachádza vo vygenerovanej množine výsledkov, test zlyhá a ohlási chybu. Testy bežia každú noc, takže v prípade výskytu chyby je jednoduché rýchlo reagovať a danú chybu odstrániť. Automatické testy nám dopomohli odhaliť pár záludných chýb.

V budúcnosti bude vývoj programu pokračovať. Tento proces sa bude sústrediť na pridanie viacerých užívateľských nastavení ako aj možné doplnenie podpory pre prácu z oboma typmi časových značiek zároveň.

Literatúra

[1] BABICA, J.: *Message Sequence Charts properties and checking algorithms*. Diplomová práca, Masarykova univerzita, Fakulta informatiky, 2011.

[2] BABICA, J.; ŘEHÁK, V.; SLOVÁK, P.; a i.: Formalisms and Tools for Design and Specification of Network Protocols. In *FIMU-RS-2007-02*, Masaryk Univerzity, 2007.

[3] BEZĎEKA, M.; BOUDA, O.; KORENČIAK, Ľ.; a i.: Sequence Chart Studio. In *Proceedings of the 12th International Conference on Application of Concurrency to System Design (ACSD'12)*, IEEE, 2012, s. 148–153.

[4] BORZA, V.: *Generation of MSC Diagrams from Network Traffic*. Bakalárska práca, Masarykova univerzita, Fakulta informatiky, 2013.

[5] DECHTER, R.; MEIRI, I.; PEARL, J.: Temporal constraint networks. Ročník 49, 1992: s. 61–95.

[6] ITU Telecommunication Standardization Sector – Study group 17: ITU Recommendation Z.120, Message Sequence Charts (MSC). 2011.

[7] KORENČIAK, Ľ.: *Effective Algorithms for Time Relation Checking in Message Sequence Charts*. Diplomová práca, Masarykova univerzita, Fakulta informatiky, 2011.

[8] KORENČIAK, Ľ.: *Time Extension of Message Sequence Chart*. Bakalárska práca, Masarykova univerzita, Fakulta informatiky, 2012.

[9] MALOTA, M.: *Layout Configuration for Message Sequence Charts*. Bakalárska práca, Masarykova univerzita, Fakulta informatiky, 2012.