Proposition Calculator in Vogelvlucht

Jesse van Son - s4601262 Aucke Bos s4591496
Milan van Stiphout s4596269 Leo Cornelissen s4606566

June 17, 2016

1 Voorwoord

Dit verslag is geschreven om een documentatie te geven van de uiteindelijke 'Proposition Calculator' app. In dit verslag zal een beschrijving van de applicatie gegeven worden, ook zullen we het nut van de applicatie uitleggen en het ontwerp nader toelichten. Hieronder is een korte inhoudsopgave te zien.

2 Beschrijving

2.1 Inleiding

Ons product is een applicatie om waardes van propositionele formules te berekenen, wat blijkt uit de productnaam 'Proposition Calculator'. De app heeft een aantal functies, zoals te zien op bijgevoegde afbeelding. Dit is het menu zoals de gebruiker te zien krijgt als de app voor het eerst geopend wordt.

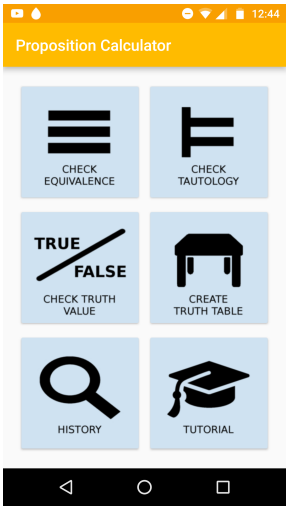

Figure 1: Main menu

1. Check Equivalence
2. Check Tautology
3. Check Truth Value
4. Create Truth Table
5. History
6. Tutorial

In de tutorial wordt beschreven hoe propositielogica werkt en wat de verschillende functies van de app inhouden. Deze functie is met name voor beginners in de propositielogica, die graag willen doorgronden hoe deze logica werkt. Als we op de 'Check Equivalence'-optie drukken, zien we de activiteit die de app gebruikt om de gebruiker een propositionele formule te laten maken. Deze activiteit wordt in alle activiteiten van de app aangeroepen om logische formules te creëren en later te parseren. Nadat bekeken is of de ingevoerde formules equivalent zijn, kan de gebruiker kiezen om verder te werken met één van de formules uit de equivalentie. Zij kan bijvoorbeeld een truth table laten generen voor de gekozen formule. Dit 'recyclen' van formules is een groot voordeel van onze app, omdat op deze manier veel typwerk de gebruiker bespaard blijft.

Tevens is er een History-functionaliteit te vinden in de app, welke alle eerder ingevoerde formules laat zien. Zodra één van deze formules aangedrukt wordt komt de gebruiker weer in het hoofdmenu. Dit wordt in de code gerepresenteerd door een aparte klasse waar wederom een bepaalde functionaliteit aangedrukt kan worden. Het voordeel van deze functionaliteit is dat de gebruiker al een oude formule gespecificeerd heeft en hiermee verder lijkt te willen werken. Als de gebruiker dus de optie voor 'Check Tautology' selecteert, berekent de app direct of de formule uit de history een tautologie is. Dit maakt de app in combinatie met de hiervoor gegeven optie in 'Check Equivalence' erg zuinig qua typwerk en dus bijzonder gebruiksvriendelijk.

2.2 Productverantwoording

Onze applicatie is erg handig voor mensen met interesse voor wiskunde en logica. De focus ligt op simpliciteit en ease-of-use, terwijl we toch een krachtige tool leveren die overal gebruikt kan worden. De app is geschikt voor de beginners maar ook voor gevorderden als rekenhulp.

We hebben maar één applicatie gevonden met ongeveer dezelfde functionaliteit, en dat was 'Logic Calculator' door Hernan Gabriel Romero. Deze app heeft minder functionaliteit dan ons eindproduct, zelfs voor betalende gebruikers. Een voorbeeld hiervan is het gebrek aan een history function. Ook is een groot nadeel dat de user interface naar onze mening minder duidelijk is. Het toetsenbord verdwijnt bijvoorbeeld niet automatisch nadat een formule ingevuld is, waarna het lastig is om te zien hoe het toetsenbord wel weg moet. Dit hoeft geen probleem te zijn maar het toetsenbord blokkeert dan de uitkomst. Een ander nadeel is dat de truth en false sliders niet

interactief zijn; de waarden van de variabelen kunnen wel aangepast worden maar dan gebeurt er niets. Pas nadat de formule opnieuw ingevoerd is werkt dit.

Om bovenstaande redenen denken wij dat onze app veel toevoegt aan de al bestaande concurrentie. Ook is er op het moment van schrijven maar één andere app met hetzelfde doel en een enigszins strakke uitwerking, en het is altijd goed om keuze en concurrentie te hebben. Dit bevordert de innovatie en houdt andere app developers ook scherp.

2.3 Specificaties

Een groot deel van de eigenschappen van onze applicatie zijn al gegeven bij de Inleiding. Als een beginnende gebruiker meer wil weten over propositielogica, dan zou hij eerst de tutorial kunnen doornemen. Hierna zal hij een van de functies willen proberen. Vaak begint een gebruiker linksboven, dan komt de gebruiker bij de check equivalence functie. Hier zou hij dan twee simpele formules in kunnen voeren om zo bekend te worden met de werking van de app en proposities. Nadat hij dit gedaan heeft kan de gebruiker direct zien dat hij door kan klikken naar bijvoorbeeld een truth table, op deze manier is het heel gemakkelijk te gebruiken voor beginnende gebruikers. Dit is dan ook een van de eigenschappen van deze applicatie

We hebben heel veel specificaties in onze app, in de inleiding is een opsomming gegeven met wat de opties zijn van onze app. Deze opties zijn een groot deel van onze specificaties. Een andere specificatie is dus het makkelijk doorklikken naar andere opties met je formule zoals hierboven genoemd is. Ook is de app gemakkelijk te gebruiken voor beginners door de tutorial en de gebruiksvriendelijkheid.

Als laatst hebben we ook nog de specificatie dat de app interactief werkt. Bijvoorbeeld bij het checken van de truth value kan je interactief schuiven met de sliders en zo zien wat de waarde van de logische formule wordt.

3 Ontwerp

3.1 Globaal ontwerp

Overal in de app hebben we een ActionBar. Deze zorgt dat het altijd duidelijk is dat er 1 activity terug kan worden gegaan, of direct teruggekeerd kan worden naar het main menu. Dit verzekert dat de gebruiker erg mobiel is binnen de app en snel en gemakkelijk acties kan overdoen.

Het Activity Options is misschien wel de belangrijkste klasse in onze app. Deze wordt gestart bij het openen van de app. Vanuit hier worden met behulp van de knoppen alle andere activies gestart. Veel verschillende activities gebruiken overeenkomende XML bestanden. Het is bij de meeste opties namelijk zo dat er een formule ingevoerd moet worden. Deze class is ook te zien in het screenshot bij de inleiding.

De meeste knoppen starten dan ook met een ander essentieel activity: EnterFormula. In Figure 2 zien we een screenshot hiervan. Deze activity wordt op verschillende manieren aangeroepen. De optie Equivalence moet namelijk 2 keer een EnterFormula aanpassen, terwijl de andere opties deze maar 1 keer aanroepen. Daarom wordt er bij het aanroepen van activities (met behulp van een Intent) vaak een String meegegeven genaamd "modus". Dit geeft aan welke, en in welke volgorde, activities aangeroepen moeten worden.

Een ander belangrijk activity is ShowResult, deze wordt namelijk gebruikt voor zowel Check Equivalence als voor Check Tautology. Deze activity laat het resultaat zien voor de optie. Dus of de 2 formules equivalent zijn, of in het andere geval, of de formule een tautology is. Verder zijn er

weinig opties bij dit activity, behalve dat het ook hier mogelijk is om te laten zien hoe de formula gesplit wordt. Deze activity gebruikt een controller met dezelfde naam. Deze controller wordt ook nog eens gebruikt bij de optie Create Truth Table, en Create Truth Value. De controller heeft verschillende constructors, omdat niet elke "Show" een lijst met variabele en waardes nodig heeft. Denk bijvoorbeeld aan een Tautology: Deze is dan en slechts dan waar als de formule True oplevert voor alle mogelijke combinaties van waardes van de variabelen, en zal dus geen HashMap<String, Boolean> nodig hebben, maar wel een lijst met variabelen. De controller berekent voor elke optie op een andere manier het resultaat, al lijken deze manieren natuurlijk wel deels op elkaar. Voor elke functionaliteit heeft deze controller dus een andere methode.

Dit brengt ons bij een volgend zeer belangrijk activity: Split-Formula. Deze splits de huidige formula op in 2 delen. Het linker deel en het rechter deel, van de operator welke het eerst wordt geëvalueerd. Daarna kan de gebruiker er voor kiezen om op 1 van deze delen te klikken en hier op "in te zoomen". Deze klik opent hetzelfde activity, maar dan met als huidige formule het deel waarop geklikt is. Zo kan de gebruiker net zo ver zoomen (en stappen terug doen) tot de formule een variabele is. Een nuttige toevoeging aan de app.

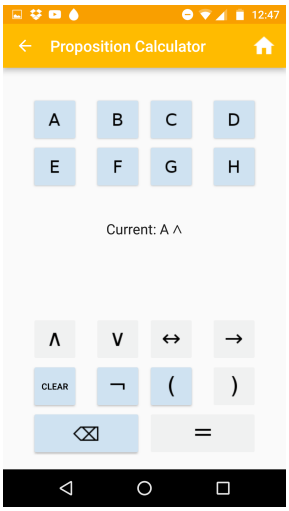

Figure 2: Enter formula

Bij de optie Check Truth Value is een belangrijk en onderscheidend element in onze app dat de values "live" aangepast kunnen worden. Er staan 8 switches, voor de 8 mogelijke variabelen. Alleen de switches welke in de formule zijn gebruikt zijn Clickable. Zodra er op een switch wordt geklikt wordt de formula opnieuw geparsed en het resultaat direct aangepast. Dit zorgt ervoor dat de gebruiker gemakkelijk verschillende combinaties van waardes kan testen op 1 formula. Bovendien zijn er ook knoppen aanwezig om in 1 klik alle switches op True danwel False te zetten.

Bij de opties Check Equivalence en Check Tautology is het bovendien mogelijk om, nadat het resultaat berekend is, een andere optie toe te passen op deze formule: Create Truth Table, of Check Truth Value. Hierbij hoeft de formule niet opnieuw ingevoerd te worden, dus het activity EnterFormula wordt overgeslagen. Dit zorgt voor een soepel en snel gebruik van de app, waarbij 1 formule meteen op verschillende manier getest kan worden.

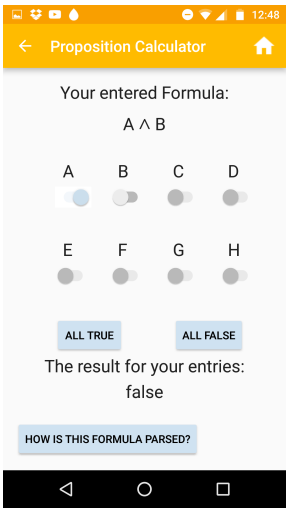

Figure 3: Truth value

Bij het activity Create Truth Table is een mooie toevoeging dat de tabel in een Horizontal ScrollView & een Vertical ScrollView staat. Bij lange formules zorgt dit ervoor dat de tabel ook naar links en rechts gescrolt kan worden, zodat de hele formule + het resultaat altijd te zien zijn. Bij formules met veel variabelen zorgt dit ervoor dat er verticaal gescrollt kan worden, waarbij de header op zijn plek blijft. Dit zorgt voor een soepele en duidelijke ervaring.

De optie History in het main menu biedt de mogelijkheid om eerder ingevule formules te herzien. Als extra hierop kan de gebruiker op een formule klikken, om hier vervolgens een andere func-

tionaliteit mee te gebruiken. Hiervoor is een nieuw activity aangemaakt: OptionsHistory. Deze gebruikt een eigen XML, welke sterk lijkt op die van het main menu. Een belangrijk verschil: De onderste 2 knoppen, History en Tutorial, zijn vervangen door 1 grote terug knop. Er is gekozen voor een nieuw activity omdat elke knop in dit menu een andere activity aanroept dan in het main menu. De formule is namelijk al bekend, dus EnterFormula kan altijd overgeslagen worden. Verder lijkt dit menu erg op het main menu en is het voor de gebruiker dus sterk herkenbaar en duidelijk in gebruik.

Het laatste functionele activity is de Tutorial. Dit is vooral handig voor gebruikers welke weinig ervaring hebben met Proposities. Hier wordt erg duidelijk en basisch uitgelegd wat proposities zijn en hoe formules geparseerd worden. We hebben ervoor gekozen deze pagina als een statische HTML pagina te maken, en deze in het activity te laden. HTML is makkelijk in het gebruik van text layout en afbeeldingen. Daarom was deze snel gemaakt met een mooie text layout en duidelijke plaatjes. Het enige wat in het activity nog moest gebeuren was deze HTML pagina in te laden. Dit is dus een zeer compact en duidelijk activity, welke toch van groot belang is voor de beginnende gebruiker.

3.2 Detailontwerp

Een essentiele variabele in de klasse EnterFormula is: LinkedList<Boolean[]> ButtonConfigHistory. Deze houdt bij wat de vorige configuratie van de knoppen was. Dit is nodig bij het gebruik van de backspace knop, knoppen moeten dan ook weer goed ingesteld worden. Als deze lijst er niet was, kunnen alle knoppen altijd ingedrukt worden, en zijn er formules in te voeren welke niet logisch zijn. Door het limiteren van in te drukken knoppen, is het voor de gebruiker niet mogelijk onlogische input te geven waardoor de app crasht. Een zeer belangrijk element van de app dus, want de invoer van de gebruiker is vaak een punt waar de app crasht.

De belangrijkste attributen in deze app zijn duidelijk de formules. Deze worden opgeslagen in in een ArrayList<String>. Aan elke intent welke 1 of meerde formules nodig heeft wordt deze meegegeven in een Bundle. We gebruiken een Bundle omdat er vaak meerdere dingen worden mee gegeven: Formule1, Formule2, HashMap<String, Boolean> map (Variabelen en hun waarde), String mode. Veel activities gebruiken een of meerdere van deze variabelen, en deze worden dus altijd meegegeven in een bundel. Er zijn activities welke toegang hebben tot variabelen die ze nooit gebruiken, maar dit is geen probleem want als ze niet gebruikt worden, worden ze ook niet onnodig aangepast.

Een belangrijke klasse welke niet in een activity zit is Parser. Deze parseert en evalueert daadwerkelijk de formule, gebruikmakend van de ArrayList<String> formula, en de HashMap<String, Boolean> map. Dit is eigenlijk de core van onze app, zonder deze klasse werkt er geen enkele functie in onze app. De belangrijkste methode: evaluate(), welke daadwerkelijk de formula evalueert.

Een klasse welke hierop lijkt is SplitFormula. Deze evalueert de formule niet, maar kijkt alleen hoe een formule gesplit wordt. Dit wordt gedaan door te kijken welke operator het minste prioriteit heeft, en dus welke het middelpunt is van het linker en rechter deel. Een formule wordt hier dus wel deels geparseerd, maar de waardes van de variabelen zijn niet van belang. HashMap<String, Boolean> map wordt hier dan ook niet gebruikt.

3.3 Ontwerpverantwoording

Voor het ontwerp hebben we gekozen voor een gelige oranje kleur, we zien deze kleur niet vaak in android apps en vonden het daarom een leuke keuze. We hebben zoals in de screenshots te zien is in de action bar onze main oranje kleur, achter de status balk is deze iets donkerder. voor de

accent color hebben we gekozen voor heel licht blauw zoals te zien is op de knoppen. Hier hebben we met veel tinten blauw gespeeld omdat dit mooi staat bij wit en geel/oranje. Uiteindelijk is na veel proberen deze kleur eruit gekomen.

Op de hoofdpagina hebben we gekozen voor zes grote en duidelijke knoppen die heel het scherm vullen, dit vonden wij een duidelijke interface en er is geen lelijke ruimte meer over op het scherm. We hebben veel geëxperimenteerd met de margins en groottes van deze knoppen en we zijn met ons eindresultaat zeer tevreden. Om deze knoppen mooi te maken hebben we in de XML een style attribute meegegeven voor button.colored, deze gebruikte dan standaard onze licht blauwe accent color. De buttons die we hier gebruiken zijn image buttons, zodat we makkelijk de plaatjes erop konden maken. De plaatjes zijn allemaal gemaakt met behulp van GIMP en hebben een transperante achtergrond en een grootte 200x200 pixels. Het style attribute wat we eerder noemden heeft ook grote voordelen tegenover bijvoorbeeld de achtergrond van de image diezelfde kleur blauw te maken in plaats van transperant. Doordat we een style gebruiken voor de buttons behouden we de mooie vormen en layout van het material design van Google, als we op een knop drukken hebben we ook nog steeds het ripple effect over de knop heen. Stel dat we de achtergrond van de kleur zouden veranderen dan zou de knop gewoon statisch een blok van de kleur worden en verder geen effecten meer vertonen. Dit is waarom wij voor een style gekozen hebben.

4 Reflectie

Over het algemeen zijn we zeer tevreden over hoe het volledige proces in ons groepje is verlopen. We zijn trots op ons resultaat, want de app ziet er strak uit en doet wat hij moet doen. We hebben besloten één feature te schrappen, maar dit kwam doordat deze niet echt heel nuttig was en was geen gevolg van een gebrek aan tijd of iets dergelijks. We hebben relatief moeilijke concepten toegepast op manieren zodat ze ook echt wat toevoegden aan onze app, niet puur om ons punt te verhogen. Een goed voorbeeld hiervan is de 'history' feature, die ook na het afsluiten en opnieuw opstarten van de app de geschiedenis laat zien. Dit blijvende geheugen is een extra moeilijkheid, maar we hebben deze toegepast omdat wij vonden dat het iets toevoegde aan de app en niet omdat het de moeilijkheid van ons werk verhoogde.

De samenwerking in onze groep ging goed. Ondanks dat het werk aan de app verschilt, is het vak meer dan alleen deze app en hebben wij ervoor gezorgd dat mensen die wat minder waren in programmeren wat meer met design, presenteren of expert reviews bezig hielden. Zo hebben we een mooie samenwerking bereikt die door iedereen als prettig ervaren werd. De rollen waren ook duidelijk voor iedereen, zodat we ons nooit last-minute in hoefden te schrijven voor presentaties of deadlines. We hebben bijzonder hard gewerkt aan alle aspecten die dit vak omvat en zijn op alle fronten zeer blij met wat we hebben bereikt. Wel hadden wij het proces van het bouwen van de app zelf eerder moeten beginnen. We waren ruim op tijd klaar dus het was uiteindelijk geen probleem, maar als we eerder dagen gekozen hadden waarop we gezamenlijk werkten, hadden we nóg eerder klaar kunnen zijn en tijd kunnen besteden aan extra features. Echter, dit was moeilijk te realiseren omdat andere vakken en sociale verplichtingen ook tijd uit ons rooster opeisten.

We zijn ook erg te spreken over wat wij hebben kunnen realiseren met Android als platform. Allen hebben wij op ons eigen interessegebied (zij het graphics of programmeren) gewerkt met Android Studio, waarin wij Activities, XML en klassen constant moesten combineren tot een goedwerkend geheel. We hebben veel designbeslissingen moeten maken en daar ook veel van geleerd. Dit heeft ons een strak inzicht gegeven in de werking van Android apps en heeft onze Java- en algemene programmeerkennis verbeterd.

Eén ding wat we de volgende keer wel beter gaan doen is meteen beginnen met de code netter

maken en documenteren. We hebben nu vrij veel tijd besteed aan het optimaliseren van huidige code en het verbeteren van nieuwe code. Als we dit meteen doen, scheelt dat ons de volgende keer een hoop tijd.