Università degli Studi di Torino

Dipartimento di Informatica

Corsi di Laurea in Informatica

Tesi di Laurea in Informatica

Sintesi vocale di formule matematiche in linguaggio naturale tramite tecniche di NLG

(CURRICULUM Sistemi per il Trattamento dell'Informazione)

Relatore:
Prof. Alessandro Mazzei

Candidato:
Michele Monticone

Sessione Aprile 2019
a.a. 2017/2018

Prefazione

La traduzione e la sintesi di formule matematiche in linguaggio naturale consiste nella generazione di una rappresentazione testuale della formula a partire dalla sua rappresentazione LaTeX.

Il metodo di riferimento per lettura delle formule matematiche da parte di chi possiede una disabilità visiva consiste nella lettura della loro rappresentazione LaTeX. La sua verbosità però rende comunque lenta, e quindi poco efficiente, la lettura delle formule. Inoltre questo approccio non consente l'accesso alle formule a coloro che non conoscono questo linguaggio.

Il problema è stato affrontato in due fasi distinte. Nella prima fase si esegue una traduzione da LaTeX , il quale è un linguaggio tipografico e quindi usato per descrivere la sintassi della formula, in *Content MathML*, ovvvero un linguaggio usato per descrivere la semantica della formula. Questa traduzione viene prima effettuata dal software *LateXML* e poi post-processata, per aumentarne la precisione in particolari contesti, attraverso delle regole ad-hoc. Nella seconda fase invece si procede alla generazione vera e propria della rappresentazione testuale della formula con l'ausilio della libreria *SimpleNLG-IT*.

Il software sviluppato in questo lavoro di tesi riesce a tradurre in maniera corretta la maggior parte delle formule matematiche più comuni. Nel lavoro di tesi si è eseguita una valutazione quantitativa e qualitativa di questo processo attraverso uno *user-test* online progettato appositamente.

La trasformazione di formule matematiche in comune testo, permette agli screen-reader di poter leggere le formule matematiche come se fossero

del normale testo. In questo modo gli utenti ipovodenti possono ottenere un migliore accesso a documenti scientifici.

Ringraziamenti

Vorrei ringraziare il Prof. Alessandro Mazzei per la sua disponibilità e per avermi guidato, con entusiasmo e passione, durante lo svolgimento di questo lavoro di tesi.

Un sentito ringraziamento anche ai membri del Laboratorio S. Polin e in particolar modo alla Prof. Anna Capietto e Cristian Bernareggi per i preziosi consigli e l'aiuto che mi hanno fornito.

Infine un ringraziamento alla mia famiglia e agli amici che mi sono stati vicini durante il mio percorso di studio.

Indice

Prefazione **iii**

1 Introduzione **1**

1.1 Obiettivi 1

1.2 Panoramica del sistema 2

1.2.1 Formato dell'Input 3

1.2.2 Rappresentazione semantica 4

1.2.3 Architettura del sistema 5

1.3 Principali problematiche 6

1.3.1 Problematiche in fase di analisi 6

1.3.2 Problematiche nella fase di generazione 8

1.4 Struttura della tesi 9

2 Analisi linguistica delle formule matematiche **11**

2.1 Cenni sulla struttura del linguaggio naturale 11

2.1.1 Alberi a costituenti 12

2.1.2 Alberi a dipendenze 14

2.1.3 L'albero di SimpleNLG 15

2.2 Frasi matematiche 17

2.3 Struttura linguistica delle formule 18

2.3.1 Operatori relazionali 19

2.3.2 Operatori aritmetici, algebrici e insiemistici 22

2.3.3 Connettivi logici 25

2.3.4 Insieme condizionale 27

2.3.5 Coppia 29
2.3.6 Aux 31
2.3.7 Sequenze 33
2.3.8 Funzioni elementari 38
2.3.9 Calculus 40

3 Sintesi delle frasi matematiche 45
3.1 Differenza tra linguaggio scritto e quello parlato 45
3.1.1 Elementi caratterizzanti del parlato 45
3.1.2 SSML 46
3.1.3 Matematica parlata 46
3.2 Aggregazione delle sottoformule 47
3.2.1 Priorità della matematica scritta 49
3.2.2 Priorità della matematica parlata 49
3.2.3 Il ruolo delle pause nella matematica parlata 50
3.2.4 Strategie di parentesizzazione 51
3.2.5 Estensione dell'aggregazione al resto degli operatori 51
3.3 Algoritmo di Aggregazione 53
3.3.1 Precedenza degli operatori 53
3.3.2 Algoritmo di aggregazione 54
3.4 Esempio di aggregazione 57

4 Design e Implementazione 59
4.1 Fase di analisi del LaTeX 59
4.1.1 Content MathML 59
4.1.2 Produzione del Content MathML 60
4.2 Fase di Generazione della Frase Matematica 63
4.2.1 L'albero linguistico 63
4.2.2 Realizzazione del parlato 66
4.2.3 Esempio di sintesi di una formula 67

5 Sperimentazione 71
5.1 Risultati 73

5.1.1 Formule facili 74
5.1.2 Formule difficili 75
5.2 Analisi dei risultati 80

6 Conclusioni **83**
6.1 Il lavoro svolto 83
6.2 Possibili lavori futuri 84

Capitolo 1

Introduzione

Molti documenti scientifici contengono elementi, come formule, grafici e codice, che li rendono non completamente accessibili per chi possiede delle disabilità visive. Questo è dovuto al modo in cui questi elementi vengono letti. Difatti vengono sintetizzati mediante la lettura diretta del LaTeX piuttosto che una loro lettura naturale.

Nel lavoro di questa tesi ci si è concentrati sull'accessibilità delle formule matematiche e si è sviluppato un software che permette la loro sintesi vocale naturale.

1.1 Obiettivi

Ai fini del lavoro di tesi ci si è posti come obiettivi:

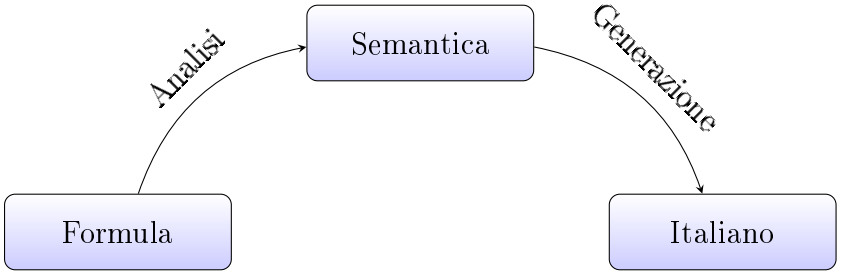
1. capire le problematiche della lettura di formule matematiche concentrandosi sulle esigenze dei non vedenti.

2. stabilire se esiste una precisa struttura linguistica dietro le formule matematiche.

3. costruire un sistema di sintesi da formula matematica a Italiano.

1.2 Panoramica del sistema

L'idea generale è quella di modellare il sistema come se fosse una traduzione interlingua fatta da un'analisi e da una generazione. Lo schema di traduzione può essere rappresentato con il triangolo di Vauquois (Hutchins [?]) come riportato di seguito.



Durante lo svolgimento del lavoro di tesi si è posta enfasi sulla fase di generazione del parlato assumendo di avere a disposizione le formule già in una rappresentazione semantica.

È stato comunque necessario effettuare la fase di analisi per eseguire prove sul sistema ma ci si è appoggiati su uno strumento preesistente, ovvero *LatexML*[?].

Il progetto è stato sviluppato in Clojure [?], un linguaggio funzionale compilato ed eseguito dalla *Java Virtual Machine (JVM)*. Ha il vantaggio di essere funzionale ma al contempo di poter interagire con ogni tipo di libreria Java. Inoltre è altamente portabile in quanto può essere eseguito su qualsiasi dispositivo munito di `JVM`, la quale è ormai disponibile quasi ovunque.

Il software realizzato è stato rilasciato come open source sotto licenza GPL ed è consultabile al link `https://bitbucket.org/tesimagistralemonticone/formula-to-speech/src/master/`.

Al momento la sintesi delle formule avviene solo in Italiano ma il sistema è stato concepito per essere multilingua. Difatti all'interno del sistema è codificata solamente la struttura linguistica delle formula mentre la lessicalizzazione italiana avviene tramite un dizionario esterno al

sistema. È quindi possibile fornire un dizionario in una lingua diversa a patto che questa ammetta la stessa struttura linguistica delle formule.

Verranno ora presentate alcune delle scelte fatte riguardanti il sistema.

1.2.1 Formato dell'Input

Per prima cosa si è dovuto scegliere quale formalismo usare per le formule. Le due opzioni principali erano LaTeX e *Presentation MathML*. Entrambi i formalismi sono orientati alla presentazione e quindi non permettono una facile estrazione della semantica.

Esempio 1.2.1. Si consideri come esempio la funzione identità $f(x) = x$. In un linguaggio orientato alla presentazione verrebbe considerata come la sequenza di simboli "*f, (, x,), =, x*". Si può notare che tale rappresentazione non indica cosa rappresentino i simboli che occorrono. Potrebbe trattarsi anche di una moltiplicazione tra la variabile f e la variabile x. In un linguaggio semantico invece la formula verrebbe considerata come la funzione f a cui è applicato il parametro x e che restituisce x.

Ai fini del sistema nessuno dei due formalismi presenta un vantaggio significativo rispetto all'altro. Si è quindi deciso di usare il LaTeXper via della sua compattezza e diffusione.

Fra i possibili casi di studio da prendere in considerazione nel corso della tesi come testi di input vi sono:

- Il libro di analisi 1 di Pandolfi reperibile dal link `http://www.integr-abile.unito.it/Libri/Analisi1/html/analisi1.html` , da qui in poi chiamato *Il pandolfi*.
- I tre volumi di *The Feynman Lectures on Physics* reperibili dal link `http://www.feynmanlectures.caltech.edu/info/`

Entrambi i libri sono liberamente consultabili e mettono a disposizione ogni formula sia in formato LaTeX che Presentation MathML. Si è scelto di usare come caso di studio il Pandolfi.

1.2.2 Rappresentazione semantica

Fra i vari formalismi di rappresentazione della semantica di una formula quelli che spiccano sono *Content MathML* e *OpenMath*.

Content MathML È un formato `XML` e permette di rappresentare la semantica tramite un insieme chiuso di tag. In altre parole la semantica di un simbolo di funzione viene rappresentato da un tag definito nel linguaggio. Risulta quindi molto semplice la fase di parsing ma al contempo non è possibile rappresentare la semantica di simboli di funzioni che non sono già presenti nella definizione del linguaggio.

Esempio 1.2.2. Ad esempio l'espressione $a + b$ può essere rappresentata in *Content MathML* come

```
<apply>
    <plus/>
    <ci>a</ci>
    <ci>b</ci>
</apply>
```

OpenMath È simile a *Content MathML* ma usa un dizionario estendibile che associa a ogni simbolo di funzione la sua semantica. Il dizionario è estendibile a piacere e quindi in linea di principio è possibile rappresentare la semantica di ogni simbolo di funzione.

Esempio 1.2.3. Ad esempio l'espressione $a + b$ può essere rappresentata in *Open Math* come

```
<OMA>
    <OMS cd="arith1" name="plus"/>
    <OMV name="a"/>
    <OMV name="b"/>
</OMA>
```

Content MathML esteso con OpenMath È anche possibile combinare i due formalismi. Content MathML infatti permette di sfruttare *OpenMath* per rappresentare i simboli di funzione non direttamente rappresentabili con *Content MathML*.

Esempio 1.2.4. Ad esempio l'espressione $a + b$ può essere rappresentata in *Content MathML* esteso con *Open Math* come

```
<apply>
    <csymbol cd="arith1">plus</csymbol>
    <ci>a</ci>
    <ci>b</ci>
</apply>
```

Come input del sistema si è scelto di usare il terzo formalismo ma con l'accortezza di trasformare i simboli di *OpenMath* in dei veri e propri tag `XML` in moda da poter trattare l'input omogeneamente.

1.2.3 Architettura del sistema

L'architettura del software è composta da 4 moduli:

LatexMl : è il modulo che ha il compito di invocare lo strumento *LatexML* passandogli in input la formula LaTeX. L'output del modulo è la prima versione di *Content MathML*.

PostProcessor : questo modulo ha il compito di post processare il *Content MathML*per renderlo compatibile con il sistema. Il suo output è la versione *trattata* di *Content MathML*.

F-T-S (formula-to-speech): è il modulo che si occupa di generare la rappresentazione testuale della formula. Il suo output è un testo.

SynthCaller : è il modulo con che ha il compito di invocare il sintetizzatore vocale e ottenere un file audio in formato `wav`.

L'architettura nella sua interezza è rappresentata in figura 1.1.

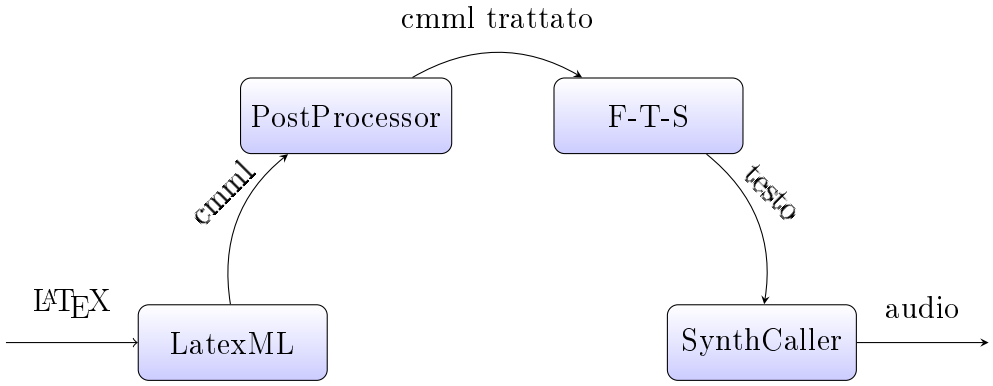


Figura 1.1: L'immagine raffigura l'architettura del software.

1.3 Principali problematiche

Le principali problematiche incontrate durante lo sviluppo del sistema riguardano sia la fase di analisi che quella di generazione. Verranno quindi discussi alcuni dei problemi più significativi riscontrati.

1.3.1 Problematiche in fase di analisi

La notazione matematica è un linguaggio usato per trasmettere conoscenza matematica da persona a persona e che quindi la rende, per certi aspetti, simile ai linguaggi naturali. Difatti in questi ultimi si sfrutta l'ambiguità per formulare messaggi brevi e coincisi, e si conta sulle conoscenze dell'interlocutore per risolvere tali ambiguità. Allo stesso modo nella notazione matematica vi sono ambiguità che vengono risolte facilmente dal lettore ma che possono creare problemi in fase di analisi. Verranno quindi presentati i principali problemi che si incontrano durante questa fase, seguendo la presentazione di Kohlhase [**?**].

Problema disambiguazione strutturale

In alcuni casi la dalla presentazione di una formula è possibile estrarre direttamente la struttura della formula. Ad esempio da

$$\frac{1}{\sqrt{2\pi}\sigma}$$

si evince facilmente, dalla lunghezza della barra, che l'argomento della radice è solamente 2π. Purtroppo non sempre ciò è possibile. Ad esempio considerando

$$\sum_{i=0}^{n} x_i + \alpha$$

in assenza di perentesi si da per scontato che solamente x_i ricada sotto la sommatoria. Considerando invece

$$\sum_{i=0}^{n} x_i + y_i$$

anche se le due sommatorie hanno la stessa struttura sintattica, si da per scontato che entrambi i termini ricadono sotto il simbolo di sommatoria in quanto y_i è in funzione dell'indice i della sommatoria.

Problema di elisione-ricostruzione

Spesso i testi matematici ammettono ambiguità oppure omettono informazioni semantiche per permettere una sintassi meno verbosa e quindi più fluida da leggere. Si fa affidamento sulla conoscenza del dominio del lettore per disambiguare o ricostruire le informazioni mancanti.

Il problema del contesto

La notazione matematica è dipendente dal contesto. Questo significa che uno stesso simbolo può assumere significati diversi in base al contesto in cui è usato. Si consideri come esempio la definizione di identità nel lambda calcolo tipato *à la Church*([**?**]):

$$\lambda x_\alpha.x =_\alpha \lambda y_\alpha.y \hat{=} I^\alpha$$

La prima e la terza occorrenza di α indicano il tipo di x e y. La seconda invece fa parte del simbolo di α-equivalenza che non ha niente a vedere con il tipo α di x e y. L'ultima occorrenza invece indica che fra tutti i combinatori di identità possibili si seleziona quello che accetta un termine di tipo α.

Un altro esempio rilevante è data dalla seguente formula:

$$\binom{n}{k}$$

In base al contesto in cui questa formula si trova potrebbe assumere almeno due diversi significati: la scelta di k elementi da n elementi totali, ovvero il coefficiente binomiale, oppure un vettore che contiene i valori n e k.

Inoltre nel caso in cui la formula si riferisca al coefficiente binomiale bisogna tenere conto che esistono diverse notazioni standard usate in vari Paesi:

$$\binom{n}{k},\ {}_nC^k,\ C^n_k,\ C^k_n$$

1.3.2 Problematiche nella fase di generazione

Una volta ottenuta la semantica di una formula bisogna trovare un modo efficacie per generare una sua rappresentazione testuale. Il compito non è banale in quanto si riscontrano problemi già con formule di piccole dimensioni. Per prima cosa bisogna chiarire quale notazione usare per descrivere semplici operazioni. Ad esempio la frazione

$$\frac{2}{3}$$

può essere letta in diversi modi:

- *due terzi*
- *due diviso tre*
- *due fratto tre*

- *due su tre*
- *due barra tre*
- *A numeratore c'è: due. A denominatore c'è tre.*

Dopodiché bisogna considerare quale sia il modo più efficacie per aggregare parti di formula. Ad esempio come si dovrebbe procedere per distinguere le due seguenti formule?

$$x + \frac{c}{y} \qquad \text{e} \qquad \frac{x+c}{y}$$

Una delle possibilità è quella di usare un meccanismo basato sulle pause per delimitare blocchi di operazioni da eseguire. Ovvero:

- $x + \frac{c}{y}$ letto come *x* <PAUSA> *più c fratto y*
- $\frac{x+c}{y}$ letto come *x più c* <PAUSA> *fratto y*

1.4 Struttura della tesi

La tesi è articolata in diversi capitoli:

Analisi linguistica delle formule matematiche : in questo capitolo si accennerà com'è strutturato il linguaggio naturale e diversi formalismi per rappresentarne la struttura. Dopodiché si procederà a dare una descrizione delle *frasi matematiche*, ovvero delle frasi in linguaggio naturale che rappresentano delle formule matematiche. Infine si definirà la struttura linguistica associata a ogni formula matematica.

Sintesi delle frasi matematiche : in questo capitolo si discuterà sulla differenza tra il linguaggio parlato e quello scritto. Si presenteranno inoltre alcuni metodi alternativi per aggregare, ovvero scegliere come parentesizzare, le formule matematiche e si concluderà presentando l'algoritmo in grado di fare ciò.

Design e Implementazione : in questo capitolo si procederà a illustrare l'architettura del software realizzato discutendo come avvengono le fasi di analisi e di generazione.

Sperimentazione : in questo capitolo verrà presentato il test effettuato per misurare l'efficacia del software realizzato e si analizzeranno i risultati.

Conclusione : in questo capitolo si trarranno le conclusione e si elencheranno alcuni dei lavori che potrebbero essere fatti in futuro.

Capitolo 2

Analisi linguistica delle formule matematiche

La notazione matematica è concepita con lo scopo di rappresentare dei concetti matematici mediante un linguaggio simbolico scritto. Spesso viene però usata nel parlato come se fosse un vero e proprio linguaggio naturale. Stabilire quali siano le similarità linguaggio matematico e linguaggio naturale è importante in quanto ciò comporterebbe l'esistenza di una struttura linguistica la quale potrebbe essere sfruttata per tecniche di generazione del linguaggio.

2.1 Cenni sulla struttura del linguaggio naturale

Il linguaggio naturale[1] può essere visto come un mezzo di comunicazione basato sulle parole. È un linguaggio ricorsivo e per questo motivo si adatta bene ad essere generato con tecniche di NLG. Questo perché la generazione viene fatta, anche, attraverso alberi sintattici che per definizione sono strutture ricorsive.

[1] con linguaggio naturale si intende le circa 5000 lingue parlate dagli uomini.

Formalmente il linguaggio naturale può essere visto come un linguaggio *tiepidamente* dipendente dal contesto (Jurafsky [**?**]) anche se per i fini di questo lavoro di tesi è più che sufficiente assumere che sia un linguaggio libero dal contesto. Esistono diversi modi per rappresentare la struttura linguistica del linguaggio naturale. Saranno presentate tre modalità: rappresentazione mediante alberi a costituenti, alberi a dipendenze e alberi *SimpleNLG*.

2.1.1 Alberi a costituenti

Sotto l'assunzione che il linguaggio naturale sia libero dal contesto, è possibile rappresentare la sua struttura mediante una grammatica libera dal contesto (`CFG`, *Context Free Grammar*). Un esempio di semplicissima, e non esaustiva, `CFG` per il linguaggio naturale è riportata nel seguente listato.

```
1     // Regole grammaticali
2     S → NP VP
3     NP → Det N | N | NP PP | Adj NP | Adv NP
4     VP → V | V NP | VP PP
5     PP → P NP
6
7     // regole lessicali
8     Det → il | lo | la | i | gli | le
9     N → cane | palla | gomma | ...
10    Adj → bravo | ...
11    Adv → molto
12    V → gioca
13    P → di | a | da | in | ...
```

Listing 2.1: Esempio di grammatica libera dal contesto per il linguaggio natiurale.

I vari non terminali che compaiono nella grammatica possono essere descritti come

S (*sentence*): indica la frase nella sua interezza.

NP (*noun phrase*): indica un sintagma nominale, ovvero una parte di frase che contiene un nome. Ad esempio "*Il bravo cane*" o "*la palla di gomma*".

VP (*verbal phrase*): indica un sintagma verbale, ovvero una parte di frase che contiene un verbo. Ad esempio "*gioca con la palla*".

PP (*prepositional phrase*): indica un sintagma preposizionale, ovvero una parte di frase che contiene una preposizione. Ad esempio "*con la palla*".

Det (*determiner*): indica un articolo.

N (*noun*): indica un nome comune.

Adj (*adjective*): indica un aggettivo.

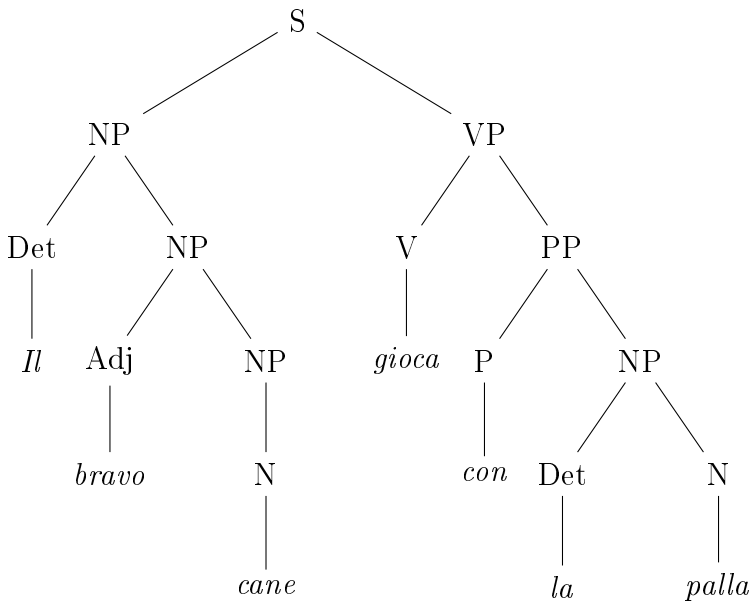
Adv (*adverb*): indica un avverbio.

V (*verb*): indica un verbo.

P (*preposition*): indica una preposizione.

Una volta definita la `CFG` è possibile sfruttarla per costruire la struttura di una frase. L'idea è quella di rappresentare la frase con un albero organizzato gerarchicamente fra gruppi di parole. Questi gruppi di parole prendono il nome di sintagmi o costituenti e l'albero così costruito prende appunto il nome di *albero a costituenti*. Nell'esempio 2.1.1 si può vedere un esempio di albero a costituenti.

Esempio 2.1.1. Si consideri ad esempio la frase "*Il bravo cane gioca con la palla*". La struttura della frase può essere espressa tramite l'albero sintattico costruito mediante la `CFG` riportata nel listato 2.1 come segue:



2.1.2 Alberi a dipendenze

Un altro modo per esprimere la struttura linguistica di una frase è quello di rappresentarla attraverso le dipendenze che intercorrono tra le varie parole. L'idea è quella di costruire un albero in cui i nodi sono le parole mentre gli archi sono le dipendenze che intercorrono tra di esse. Le dipendenze vengono indicate con dei ruoli tematici e fra i più significativi ci sono

subj : indica il soggetto.

obj : indica l'oggetto, usato quando il verbo è transitivo.

indirect-obj : indica l'oggetto indiretto, usato quando il verbo è ditransitivo.

pre-modifier : indica un modificatore che precede una parola, può essere un aggettivo o un avverbio.

post-modifier : indica un modificatore che segue una parola, può essere un aggettivo o un avverbio.

complement : indica un generico complemento.

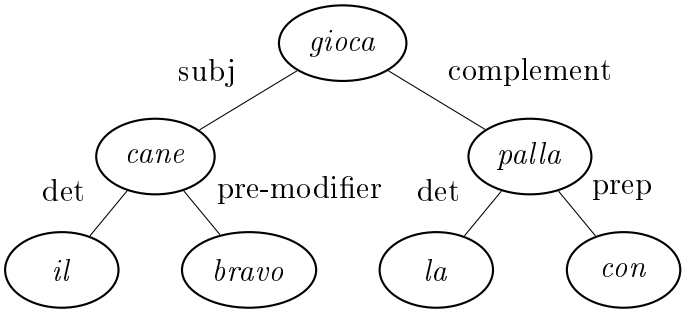
coordinate : indica una coordinazione tra due parole o frasi.

det : indica un articolo.

prep : indica una preposizione.

conj : indica una congiunzione.

Esempio 2.1.2. Si consideri la stessa frase di esempio "*Il bravo cane gioca con la palla*" usata nell'esempio 2.1.1. La struttura della frase può essere espressa tramite l'albero delle dipendenze come segue:



2.1.3 L'albero di SimpleNLG

Definire la struttura delle frasi matematiche è necessario per poter generare il linguaggio. La fase di generazione sfrutta una libreria software chiamata *SimpleNLG-IT* [**?**] che ha il compito di riordinare le parole nella frase e di eseguire la flessione morfologica. È un porting italiano della libreria software *SimpleNLG* [**?**]. Si procederà a darne una descrizione dettagliata nel capitolo 4. Al momento è sufficiente discutere sul tipo di albero che viene usato dalla libreria. Può essere visto come un albero a dipendenze che però indica le dipendenze che intercorrono tra sintagmi invece che tra le singole parole. I sintagmi usati sono

Clause : indica un'intera frase, contenente il soggetto, il verbo ed eventualmente un oggetto diretto e uno indiretto

NP : sintagma nominale, indica una frase contenente un nome.

VP : sintagma verbale, indica una frase contenente un verbo.

PP : sintagma preposizionale, indica una frase contenente una preposizione.

AdjP : sintagma aggettivale, indica una frase contenente un aggettivo.

AdvP : sintagma avverbiale, indica una frase contenente un avverbio.

Coord : indica una coordinazione di due sintagmi.

Si può notare che sono assenti i non terminali `N,V,Det, ...` in quanto vengono già considerati parte del sintagma corrispondente. Ad esempio il sintagma `NP` accetta una coppia di stringhe, ovvero l'articolo e il nome, senza bisogno di ulteriori variabili.

La dipendenze che intercorrono fra i vari sintagmi sono

subj : indica il soggetto.

obj : indica l'oggetto, usato quando il verbo è transitivo.

indirect-obj : indica l'oggetto indiretto, usato quando il verbo è ditransitivo.

pre-modifier : indica un modificatore che precede una parola, può essere un aggettivo o un avverbio.

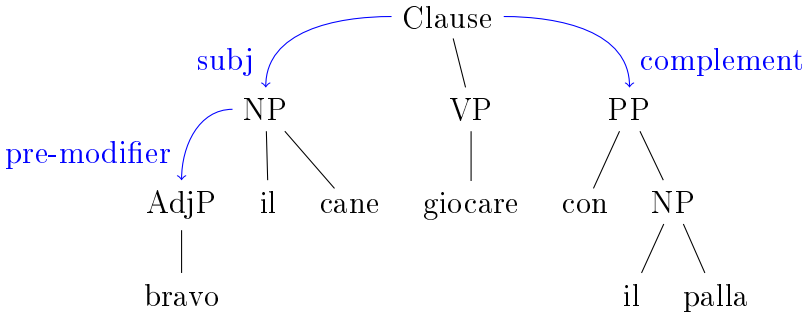
post-modifier : indica un modificatore che segue una parola, può essere un aggettivo o un avverbio.

complement : indica un generico complemento.

conj : indica una congiunzione.

coordinate : indica una coordinazione.

Esempio 2.1.3. Si consideri la stessa frase di esempio "*Il bravo cane gioca con la palla*" usata nell'esempio 2.1.1. La struttura della frase può essere espressa tramite l'albero `SimpleNLG` come segue:



Gli archi in nero rappresentano le relazioni sintagmatiche mentre in blue quelle di dipendenza.

Dall'esempio 2.1.3 si nota che viene usato sempre l'articolo *il* e il verbo è nella sua forma infinita. Ciò è dovuto al fatto che durante la costruzione dell'albero non si è ancora a conoscenza del tempo dei verbi o del genere e del numero delle parole. Sarà compito della libreria per la generazione effettuare successivamente le giuste variazioni morfologiche.

2.2 Frasi matematiche

Le formule matematiche vengono spesso pronunciate come normali frasi del linguaggio naturale. Si è scelto di assegnargli il nome di *frasi matematiche*.

In diversi casi sembra che le frasi matematiche abbiano la stessa struttura che si ritrova nel linguaggio naturale. Ad esempio una frase come "*il valore assoluto di x*" sembra essere una frase dichiarativa formata da un sintagma nominale e uno proposizionale; una frase come "*x appartiene ad a*" sembra invece essere una frase composta da un sintagma nominale e uno verbale.

Esistono però casi in cui la struttura delle frasi matematiche è abbastanza diversa da quella che normalmente si ritrova nel linguaggio naturale. Ad esempio "*due più tre*" sembrerebbe a prima vista una frase dove "*due*" è il soggetto e "*tre*" è l'oggetto. Ci si aspetterebbe quindi una struttura *soggetto-verbo-oggetto* ma "*più*" è invece un avverbio nel linguaggio naturale. Considerando però che il "*più*" si trova in una frase matematica e in questo contesto indica l'azione di sommare una quantità ad un'altra, si è scelto di considerarlo come un verbo. Lo stesso discorso viene applicato anche ad altri operatori, come si vedrà più avanti alla sezione 2.3.1.

Questa assunzione più altre che verranno illustrate più avanti permetteranno alle frasi matematiche di essere trattate allo stesso modo del linguaggio naturale.

2.3 Struttura linguistica delle formule

La struttura linguistica di una formula è costruita a partire dalla struttura degli operatori che la compongono mentre numeri e variabili sono trattati come semplici nomi.

Gli operatori vengono in genere indicati con il loro nome in *Content MathML*. Nei rari casi in cui questo nome non esiste [2] si procede ad associargli un nome significativo. In tabella 2.1 si possono vedere alcuni degli operatori considerati dal sistema e non presenti direttamente in Content MathMl.

Gli operatori presi in esame possono essere suddivisi in categorie accomunate dalla stessa struttura linguistica:

- 2.3.1 operatori relazionali
- 2.3.2 operatori aritmetici, algebrici e insiemistici
- 2.3.3 connettivi logici

[2] gli operatori rappresentati unicamente in *Open Math* ne sono un esempio.

Operatore Mancante	Nome Assegnato
$\ll$	`ll`
$\gg$	`gg`
$\{\cdots \mid \ldots\}$	`conditional-set`
$\Leftrightarrow$	`iff`

Tabella 2.1: La tabella indica alcuni dei simboli che non sono presenti in *Content MathML* e il nome che si è deciso di assegnargli.

- 2.3.4 insieme condizionale
- 2.3.5 coppia
- 2.3.6 operatori ausiliari
- 2.3.7 sequenze
- 2.3.8 funzioni elementari
- 2.3.9 calculus

Si procederà a descrivere ogni categoria, indicando quali operatori vi fanno parte e quale sia la sua struttura linguistica.

2.3.1 Operatori relazionali

Gli operatori relazionali descrivono la relazione che intercorre tra due formule e sono sono riportati nella tabella 2.2.

Si prenderanno in esame alcuni esempi significativi di frasi matematiche per comprendere la loro struttura linguistica:

1. "*x è molto maggiore di 0*"
2. "*x è minore o uguale a 0*"
3. "*A è incluso in B*"

Operatore	**Simbolo**	**Linguaggio naturale**
`gt`	$>$	è maggiore di
`geq`	$\geq$	è maggiore o uguale a
`gg`	$\gg$	è molto maggiore di
`lt`	$<$	è minore di
`leq`	$\leq$	è maggiore o uguale a
`ll`	$\ll$	è molto minore
`eq`	$=$	è uguale a
`neq`	$\neq$	non è uguale
`prsubset`	$\subset$	è propriamente incluso in
`notprsubset`	$\not\subset$	non è propriamente incluso in
`subset`	$\subseteq$	è incluso o coincidente con
`notsubset`	$\not\subseteq$	non è incluso o coincidente con

Tabella 2.2: La lista degli operatori relazionali.

Per comprendere la struttura linguistica di questo operatore si procede in prima analisi a distinguere le parti del discorso. Risulta quindi che

1. "(x, N) ($è$, V) ($molto$, Adv) ($maggiore$, Adj) (di, P) (0, N)"
2. "(x, N) ($è$, V) ($minore$, Adj) (o, Conj) ($uguale$, Adj) (a, P) (0, N)"
3. "(A, N) ($è$, V) ($incluso$, Adj) (in, P) (B, N)"

Ciò suggerisce che questa categoria di operatori può essere generata dalla seguente grammatica libera dal contesto:

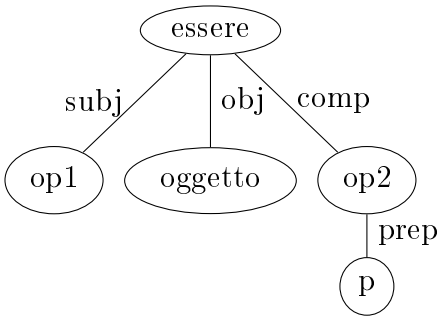
```
1    S → NP VP
2    VP → V AdjP PP
3    AdjP → Adj | Coord | Adv AdjP
4    Coord → Adj Conj Adj
```

```
5       NP → Det N
6       PP → P NP
```

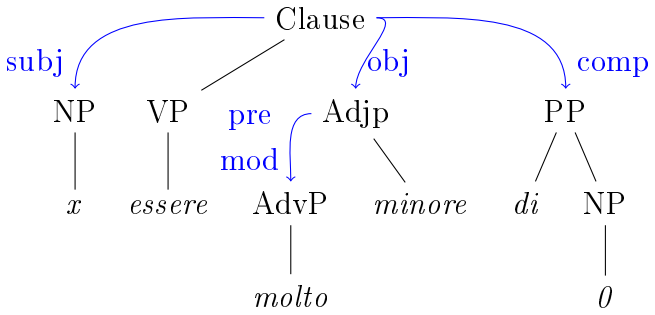
Inoltre è possibile distinguere che le frasi matematiche contenenti operatori relazionali sono composte da

- un operando `op1` a sinistra dell'operatore.
- il verbo essere che regge l'operatore.
- un `oggetto` che è un aggettivo e che può essere eventualmente coordinato con un altro aggettivo o preceduto da un avverbio.
- un altro operando `op2` a destra dell'operatore.
- una preposizione `p` che lega l'operatore al secondo operando.

Le relazioni che intercorrono fra i vari sintagmi possono essere schematizzati con il seguente albero delle dipendenze:



Esempio 2.3.1. Nel seguente albero *SimpleNLG* viene mostrato come esempio la frase "*x è molto minore di 0*".



2.3.2 Operatori aritmetici, algebrici e insiemistici

Sono gli operatori usati per rappresentare le operazioni aritmetiche, algebriche e insiemistiche. Tutti gli operatori ammettono la forma binaria infissa; inoltre la somma e la sottrazione ammettono anche quella unaria prefissa (es: -1, +2, ...). Un elenco completo di questi operatori è riportato nella tabella 2.3.

Operatore	**Simbolo**	**Linguaggio naturale**
`plus`	$+$	più
`minus`	$-$	meno
`times`	$*$	per
`divide`	$/$	diviso
`compose`	$\circ$	composto
`power`	$[\ldots]^{[\ldots]}$	elevato a
`in`	$\in$	appartiene a
`notin`	$\notin$	non appartiene a
`intersect`	$\cap$	intersecato con
`union`	$\cup$	unito a
`setdiff`	$\setminus$	differenza insiemistica
`cartesianproduct`	$\times$	prodotto cartesiano

Tabella 2.3: La lista degli operatori aritmetici, algebrici e insiemistici.

Si prendano come esempi significativi le due seguenti frasi matematiche:

1. "*meno x*"
2. "*a più b*"
3. "*a non appartiene a b*"

In base alle considerazioni fatte alla sezione 2.2 si assume che ognuno di questi operatori sia considerato un verbo. È quindi possibile distinguere le seguenti parti del discorso:

1. "(*meno*, Adj) (*x*, N) "
2. "(*a*, N) (*più*, V) (*b*, N)"
3. "(*a*, N) (*non*, Adv) (*appartiene*, V) (*a*, P) (*b*, N)"

Ciò suggerisce che questa categoria di operatori può essere generata dalla seguente grammatica libera dal contesto:

```
1    S → NP VP | NP
2    VP → V NP | Adv VP
3    NP → N | Adj N
4    PP → P NP
```

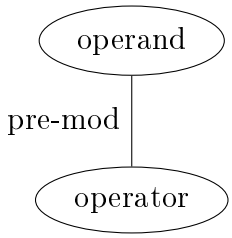
La struttura di questi operatori varia in base alla loro arietà.

Forma unaria

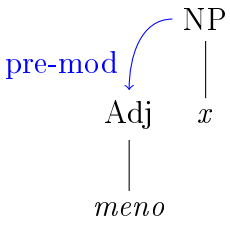
Nel caso in cui l'operatore sia in forma unaria è possibile osservare che la frase matematica di esempio è una semplice frase dichiarativa composta da

- un operando `operand`.
- un operatore `operator` che precede l'operando.

Le relazioni che intercorrono fra i vari sintagmi possono essere schematizzati con il seguente albero delle dipendenze:



Esempio 2.3.2. Nel seguente albero *SimpleNLG* viene mostrato come esempio la frase "*meno x*".

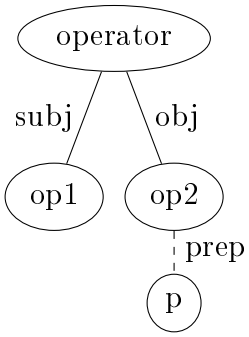


Forma binaria

Nel caso in cui l'operatore sia in forma binaria è possibile osservare che le frasi matematiche di esempio sono composte da

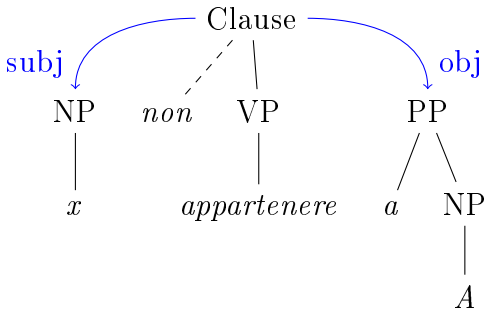
- un operando `op1` alla sinistra dell'operatore.
- un operatore `operator`.
- un operando `op2` alla destra dell'operatore.
- un eventuale proposizione `p` che lega il secondo operando con l'operatore.

Le relazioni che intercorrono fra i vari sintagmi possono essere schematizzati con il seguente albero delle dipendenze:



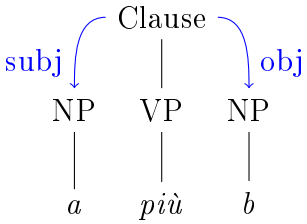
Si può notare che l'arco della preposizione è tratteggiato in quanto è una dipendenza opzionale.

Esempio 2.3.3. Nel seguente albero *SimpleNLG* viene mostrato come esempio la frase "*x non appartiene a A*".



Si può notare che l'avverbio di negazione *non* è tratteggiato per indicare che è un elemento opzionale.

Esempio 2.3.4. Nel seguente albero *SimpleNLG* viene mostrato come esempio la frase "*a più b*".



2.3.3 Connettivi logici

Sono gli operatori usati per connettere sottoformule logiche. La lista completa degli operatori logici è mostrata in tabella 2.4.

Si prendano come esempi significativi le due seguenti frasi matematiche:

1. "*A o B* "
2. "*se A allora B* "

Dall'analisi delle parti del discorso risulta che

Operatore	**Simbolo**	**Linguaggio naturale**
`and`	$\wedge$	e
`or`	$\vee$	o
`not`	$\neg$	non
`implies`	$[\varphi] \implies [\psi]$	se $[\varphi]$ allora $[\psi]$
`iff`	$\iff$	se e solo se

Tabella 2.4: La lista degli operatori logici.

1. "(A, N) (o, Conj) (B, N)"

2. "(se, Conj) (A, N) ($allora$, Conj) (B, N)"

Ciò suggerisce che questa categoria di operatori può essere generata dalla seguente grammatica libera dal contesto:

```
1    S → NP | ...
2    Coord → S Conj S | Conj S Conj S | ...
3    NP → N | ...
```

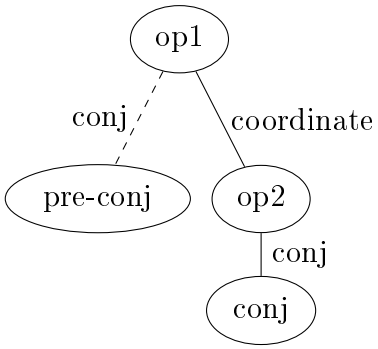
Inoltre è possibile distinguere che la frasi matematiche contenenti questi operatori sono composte da

- una congiunzione principale `conj`.

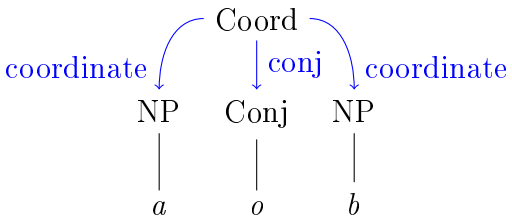
- un operando `op1` coordinato con un altro operando `op1` mediante la congiunzione principale.

- un'eventuale altra congiunzione `pre-conj` che precede la coordinazione.

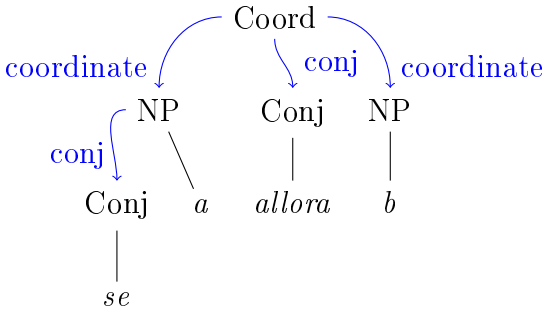
Le relazioni che intercorrono fra i vari sintagmi possono essere schematizzate con il seguente albero delle dipendenze:



Esempio 2.3.5. Nel seguente albero *SimpleNLG* viene mostrato come esempio la frase "*a o b*".



Esempio 2.3.6. Nel seguente albero *SimpleNLG* viene mostrato come esempio la frase "*se a allora b*".



2.3.4 Insieme condizionale

Questa categoria contiene un unico operatore usato per definire un insieme condizionale ed è mostrato nella tabella 2.5.

Si prenda come esempio significativo la seguente frase matematica:

Operatore	**Simbolo**	**Linguaggio naturale**
`conditional-set`	$\{[vars] \mid [cond]\}$	L'insieme di [vars] tali che [cond]

Tabella 2.5: L'operatore che rappresenta l'insieme condizionale.

1. "*L'insieme degli x tali che x è minore di 0*"

Dall'analisi delle parti del discorso risulta che

1. "(*Il*, Det) (*insieme*, N) (*di*, Prep) (*gli*, Det) (*x*, N) (*tali che*, Prep) (*x*, N) (*è*, V) (*minore*, Adj) (*di*, Prep) (*0*, N)"

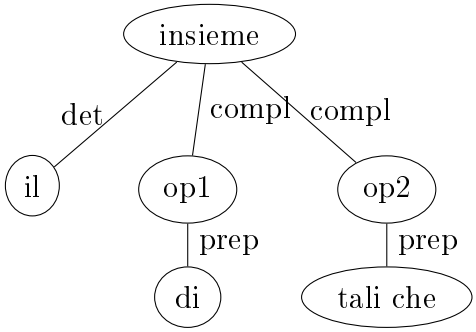
Ciò suggerisce che questa categoria di operatori può essere generata dalla seguente grammatica libera dal contesto:

```
1    S → NP VP| ...
2    NP → Det N | NP PP |...
3    VP → V PP |...
4    PP → P NP | P S |...
```

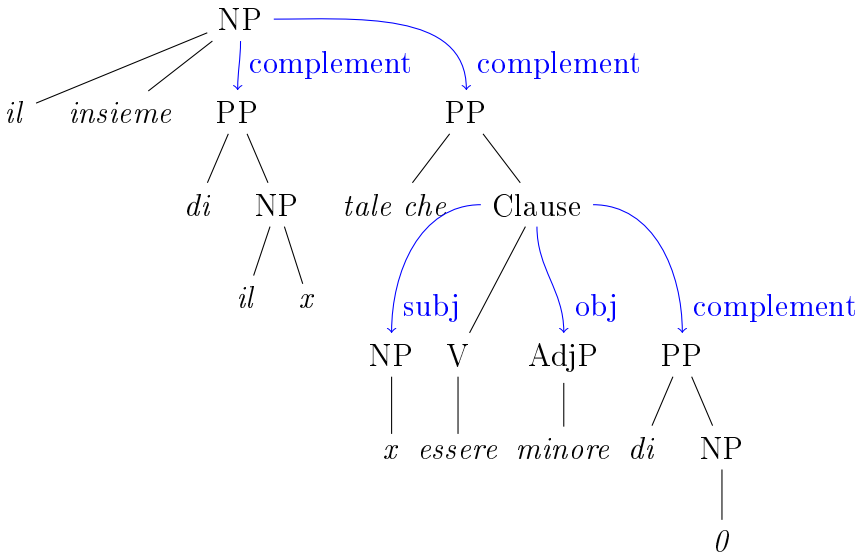
Inoltre è possibile distinguere che le frasi matematiche contenenti questo operatore sono composte da

- l'articolo "*il*".
- il nome "*insieme*".
- la preposizione "*di*".
- un operando `op1` che rappresenta gli elementi dell'insieme, ovvero una semplice variabile o un'altra frase matematica.
- la preposizione `tale che`
- un operando `op2` che rappresenta la condizione posta sugli elementi dell'insieme, ovvero una frase matematica.

Le relazioni che intercorrono fra le varie parole possono essere schematizzate con il seguente albero delle dipendenze:



Esempio 2.3.7. Nel seguente albero *SimpleNLG* viene mostrato come esempio la frase "*L'insieme degli x tali che x è minore di 0*".



2.3.5 Coppia

Questa categoria contiene un unico operatore usato per definire una coppia di elementi ed è mostrato nella tabella 2.6.

Si prenda come esempio significativo la seguente frase matematica:

1. "*La coppia di x e y*"

Operatore	**Simbolo**	**Linguaggio naturale**
`pair`	$([x], [y])$	la coppia di [x] e [y]

Tabella 2.6: L'operatore che rappresenta la coppia.

Dall'analisi delle parti del discorso risulta che

1. "(*La*, Det) (*coppia*, N) (*di*, Prep) (x, N) (e, Conj) (y, N)"

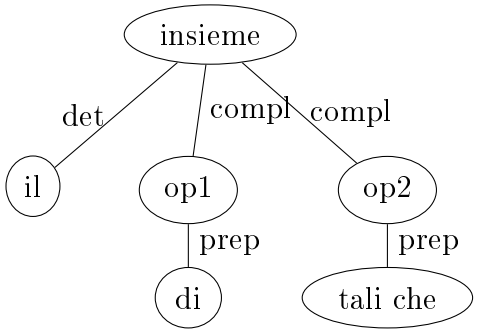
Ciò suggerisce che questa categoria di operatori può essere generata dalla seguente grammatica libera dal contesto:

```
1    NP → Det N | NP PP |...
2    PP → P Coord |...
3    Coord → N Conj N |...
```

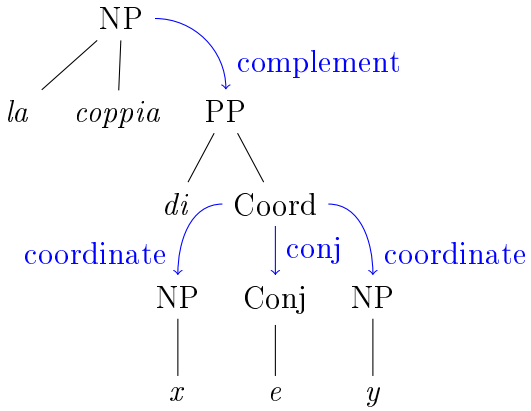
Inoltre è possibile distinguere che la frasi matematiche contenenti questo operatore sono composte da

- l'articolo "*la*".
- il nome "*coppia*".
- la preposizione "*di*".
- un operando `op1` coordinato con un altro operatore `op2` tramite la congiunzione `conj`.

Le relazioni che intercorrono fra le varie parole possono essere schematizzate con il seguente albero delle dipendenze:



Esempio 2.3.8. Nel seguente albero *SimpleNLG* viene mostrato come esempio la frase "*La coppia di x e y*".



2.3.6 Aux

Questa categoria contiene degli operatori ausiliari usati da altri operatori. La lista completa è mostrata nella tabella 2.7.

Operatore	Simbolo	Linguaggio naturale
`lowlimit`	$\sum_{i=[a]}, \prod_{i=[a]}, \int_{[a]}$	da [a]
`lowlimit`	$\lim_{x \to [a]}$	a [a]
`uplimit`	$\sum^{[b]}, \prod^{[a]}, \int^{[b]}$	a [b]
`subscript`	$[x]_{[n]}$	con [n]

Tabella 2.7: La lista degli operatori ausiliari.

Si può notare che l'operatore `lowlimit` si comporta diversamente a seconda dell'operatore principale a cui è associato.

Si prendano come esempio significativo le seguenti frasi matematiche:

1. "*x con 0*"

2. "*i da 0*"

Dall'analisi delle parti del discorso risulta che

1. "(*x*, N) (*con*, Prep) (*0*, N)"

2. "(*i*, N) (*da*, Prep) (*0*, N)"

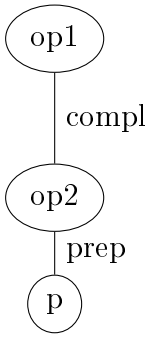
Ciò suggerisce che questa categoria di operatori può essere generata dalla seguente grammatica libera dal contesto:

```
1    NP → NP PP | N | ...
2    PP → P N |...
```

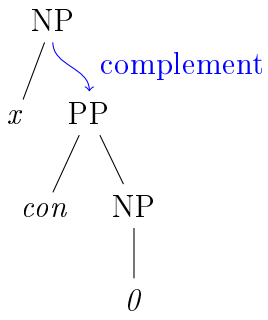
Inoltre è possibile distinguere che la frasi matematiche contenenti questo operatore sono composte da

- un operando `op1`.
- una preposizione `p`.
- un altro operando `op2`.

Le relazioni che intercorrono fra le varie parole possono essere schematizzate con il seguente albero delle dipendenze:



Esempio 2.3.9. Nel seguente albero *SimpleNLG* viene mostrato come esempio la frase "*x con 0*".



2.3.7 Sequenze

Questa categoria contiene gli operatori che rappresentano delle sequenze di valori o operazioni.

Operatore	Simbolo	Linguaggio naturale
`sum`	$\sum_{[x=\ldots]}$	la sommatoria per [x] ...
`prod`	$\prod_{x=\ldots}$	la produttoria per [x] ...
`limit`	$\lim_{x\to\ldots}$	il limite per [x] tendente ...

Tabella 2.8: La lista degli operatori che rappresentano delle sequenze di valori o operazioni

Si prendano come esempio significativo le seguenti frasi matematiche:

1. "*il limite di f di x*"
2. "*il limite per x tendente a 0 di f di x*"
3. "*la sommatoria per i da 0 a n di f di x*"

Dall'analisi delle parti del discorso risulta che

1. "(*il*, Det) (*limite*, N) (*di*, Prep) (*f*, N) (*di*, Prep) (*x*, N)"

2. "(*il*, Det) (*limite*, N) (*per*, Prep) (*x*, N) (*tendente*, V) (*a*, Prep) (*0*, N) (*di*, Prep) (*f*, N) (*di*, Prep) (*x*, N)"

3. "(*la*, Det) (*sommatoria*, N) (*per*, Prep) (*i*, N) (*da*, Prep) (*0*, N) (*a*, Prep) (*n*, N) (*di*, Prep) (*f*, N) (*di*, Prep) (*x*, N)"

Ciò suggerisce che questa categoria di operatori può essere generata dalla seguente grammatica libera dal contesto:

```
1    NP → Det N | NP PP | N | ...
2    PP → P NP | P S | ...
```

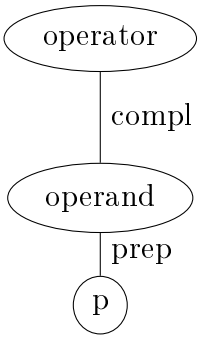
La struttura di questi operatori varia in base alla loro arietà.

Forma unaria

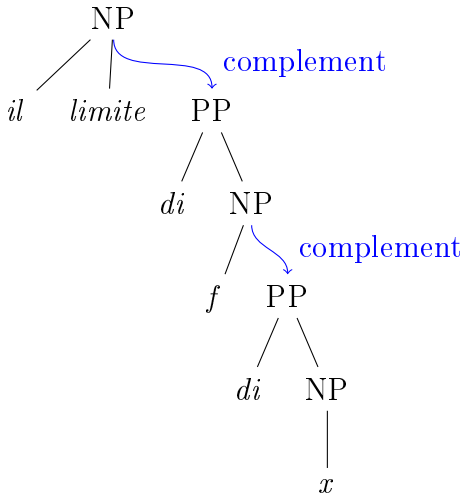
Nel caso in cui l'operatore sia in forma unaria (frase esempio 1) è possibile osservare che la frase matematica di esempio è composta da

- un operatore `operator`.
- un operando `operand` rappresentante l'espressione a cui è applicato l'operatore.
- una preposizione `p` che lega l'operatore all'operando.

Le relazioni che intercorrono fra le varie parole possono essere schematizzate con il seguente albero delle dipendenze:



Esempio 2.3.10. Nel seguente albero *SimpleNLG* viene mostrato come esempio la frase "*il limite di f di x*".



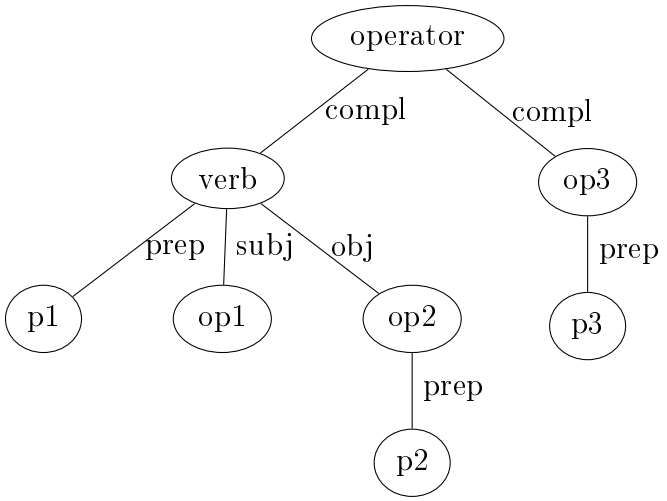
Forma ternaria

Nel caso in cui l'operatore sia in forma ternaria (frase esempio 2) è possibile osservare che la frase matematica di esempio è composta da

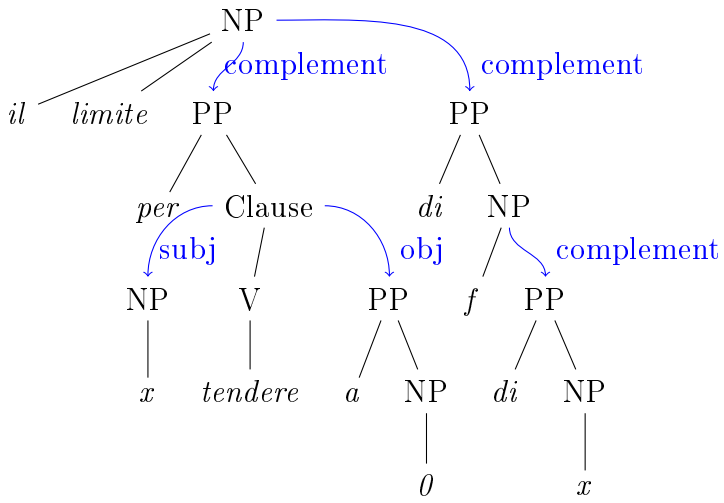
- un opertore `operator`.
- un operando `op1` rappresentante la variabile legata.
- una preposizione `p1` che lega la variabile legata.
- un operando `op2` rappresentante il limite inferiore.
- un verbo `verb` un verbo che indica la relazione tra la variabile legata e il limite inferiore, come ad esempio "*x tendente a più infinito*" o "*x appartenente ad A*".
- una preposizione `p2` che lega il verbo al limite.
- un operando `op3` rappresentante l'espressione a cui è applicato l'operatore.

- una preposizione `p3` che lega l'operatore all'espressione su cui è applicato.

Le relazioni che intercorrono fra le varie parole possono essere schematizzate con il seguente albero delle dipendenze:



Esempio 2.3.11. Nel seguente albero *SimpleNLG* viene mostrato come esempio la frase "*il limite per x tendente a 0 di f di x*".

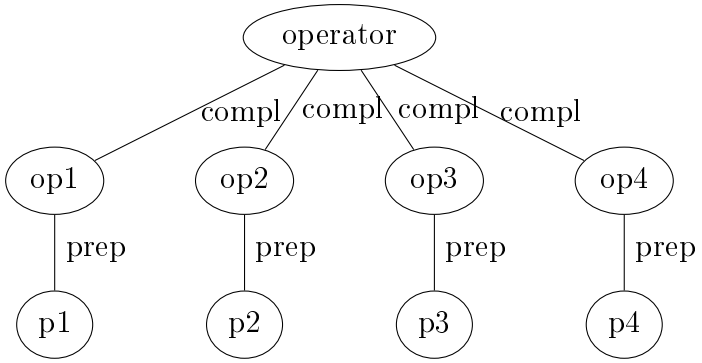


Forma quaternaria

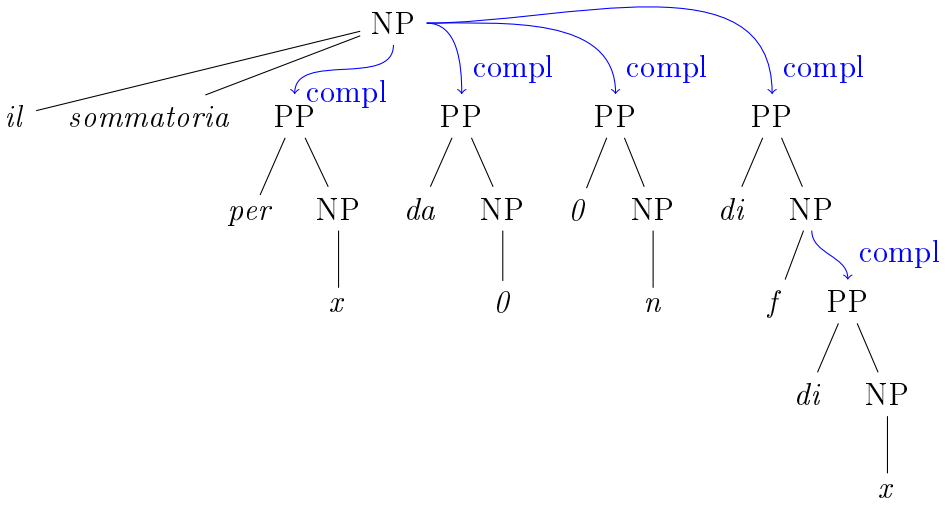
Nel caso in cui l'operatore sia in forma quaternaria (frase esempio 3) è possibile osservare che la frase matematica di esempio è composta da

- un opertore `operator`.
- un operando `op1` rappresentante la variabile legata.
- una preposizione `p1` che lega la variabile legata all'operatore.
- un operando `op2` rappresentante il limite inferiore.
- una preposizione `p2` che lega la variabile legata con il limite inferiore.
- un operando `op3` rappresentante il limite superiore.
- una preposizione `p3` che lega il limite inferiore a quello superiore.
- un operando `op4` rappresentante l'espressione a cui è applicato l'operatore.
- una preposizione `p4` che lega l'operatore all'espressione su cui è applicato.

Le relazioni che intercorrono fra le varie parole possono essere schematizzate con il seguente albero delle dipendenze:



Esempio 2.3.12. Nel seguente albero *SimpleNLG* viene mostrato come esempio la frase "*la sommatoria per i da 0 a n di f di x*".



2.3.8 Funzioni elementari

Questa categoria contiene tutte le funzioni unarie prefisse. L'elenco completo degli operatori è riportato nella tabella 2.9.

Si prendano come esempio significativo le seguenti frasi matematiche:

1. "*il valore assoluto di x*"
2. "*il seno di x*"

Dall'analisi delle parti del discorso risulta che

1. "(*il*, Det) (*valore*, N) (*assoluto*, Adj) (*di*, Prep) (*x*, N)"
2. "(*il*, Det) (*seno*, N) (*di*, Prep) (*x*, N)"

Ciò suggerisce che questa categoria di operatori può essere generata dalla seguente grammatica libera dal contesto:

```
1    NP → Det N | NP PP | NP Adj | N | ...
2    PP → P NP | ...
```

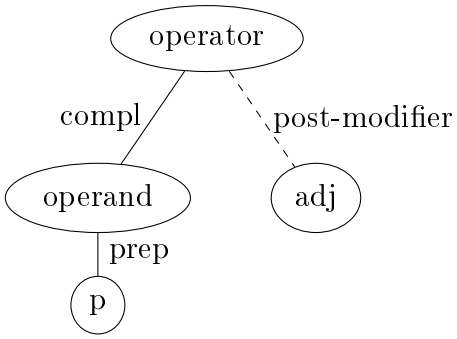
Inoltre è possibile distinguere che la frasi matematiche contenenti questo operatore sono composte da

Operatore	**Simbolo**	**Linguaggio naturale**
`sin`	sin	il seno di
`cos`	cos	il coseno di
`tan`	tan	la tangente di
`arcosin`	arcsin	l'arcoseno di
`arcosin`	arccos	l'arcocoseno di
`arcotan`	arctan	l'arcotangente di
`abs`	$\|\dots\|$	il valore assoluto di
`root`	$\sqrt[n]{[x]}$	la radice n-esima di
`factorial`	$n!$	il fattoriale di
`inverse`	f^{-1}	l'inversa di

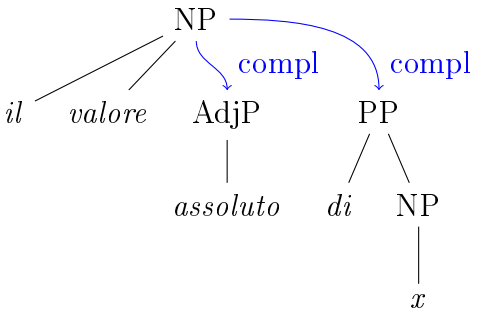
Tabella 2.9: La lista delle funzioni classiche di base

- un operatore `operator`.
- un operando `operand` rappresentante l'espressione a cui si applica l'operatore.
- una preposizione `p` che lega l'operando all'operatore.
- un eventuale aggettivo `adj` che segue l'operatore.

Le relazioni che intercorrono fra le varie parole possono essere schematizzate con il seguente albero delle dipendenze:



Esempio 2.3.13. Nel seguente albero *SimpleNLG* viene mostrato come esempio la frase "*il valore assoluto di x*".



2.3.9 Calculus

Questa categoria contiene derivate e integrali e sono mostrati in tabella 2.10.

Operatore	Simbolo	Linguaggio naturale
`int`	$\int_{[a]}^{[b]}[expr]d[x]$	l'integrale [eventuali limiti] di [expr] in de [x]
`diff`	$f^{(n)}(x), \frac{d^n f(x)}{dx^n}$	la derivata [eventuale grado n] di [f] rispetto [x]

Tabella 2.10: La lista degli operatori che rappresentano derivate e integrali

Si prendano come esempio significativo le seguenti frasi matematiche:

1. "*l'integrale di f di x in de x*"
2. "*la derivata seconda di f di x rispetto x*"
3. "*l'integrale da b a c di f di x in de x*"

Nelle frasi matematiche contenenti gli integrali viene usata la parola *de* per indicare la variabile di integrazione. Questa parola però non esiste nel linguaggio naturale. Dato che precede sempre la variabile di integrazione e in un certo senso modifica il significato, si è scelto di considerarla come un aggettivo. Risulta quindi dall'analisi delle parti del discorso che

1. "(*il*, Det) (*integrale*, N) (*di*, Prep) (*f*, N) (*di*, Prep) (*f*, x) (*in*, Prep) (*de*, Adj) (*x*, N) "
2. "(*la*, Det) (*derivata*, N) (*seconda*, Adj) (*di*, Prep) (*f*, N) (*di*, Prep) (*f*, x) (*rispetto*, Prep) (*x*, N) "
3. "(*il*, Det) (*integrale*, N) (*da*, Prep) (*b*, N) (*a*, Prep) (*c*, N) (*di*, Prep) (*f*, N) (*di*, Prep) (*f*, x) (*in*, Prep) (*de*, Adj) (*x*, N) "

Ciò suggerisce che questa categoria di operatori può essere generata dalla seguente grammatica libera dal contesto:

```
1    NP → Det N | NP PP | NP Adj | N | ...
2    PP → P NP | ...
```

La struttura di questi operatori varia in base alla loro arietà.

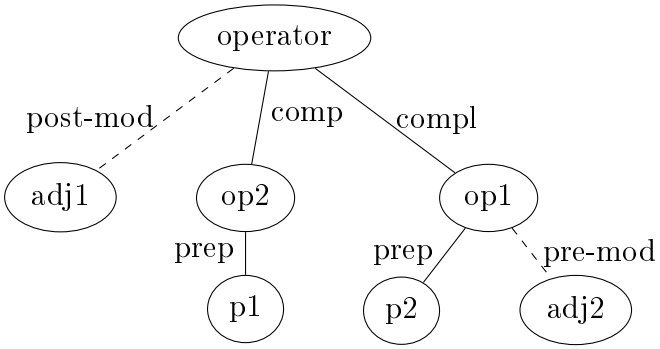
Forma binaria

Nel caso in cui l'operatore sia in forma binaria (esempio frasi 1 e 2) è possibile osservare che le frasi matematiche sono composte da

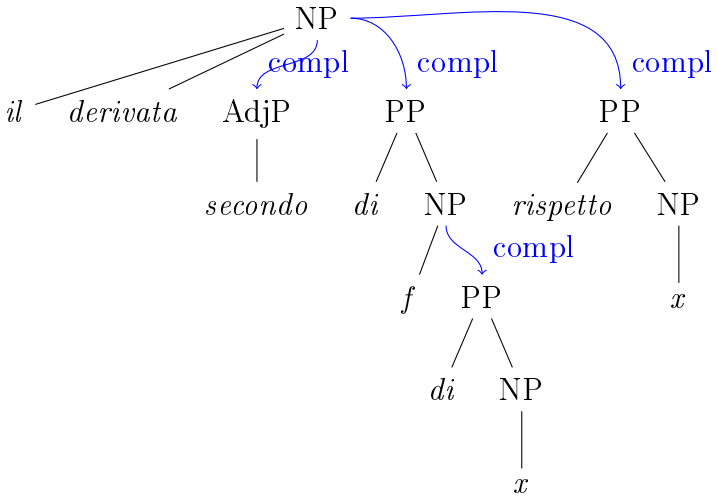
- un operatore `operator`.
- un operando `op1` rappresentante la variabile legata.

- un operando `op2` rappresentante l'espressione a cui si applica l'operatore.
- una preposizione `p1` che lega l'operatore con espressione.
- una preposizione `p2` che lega l'espressione con la variabile legata.
- un eventuale aggettivo `adj1` che segue l'operatore.
- un eventuale aggettivo `adj2` che precede la variabile legata.

Le relazioni che intercorrono fra le varie parole possono essere schematizzate con il seguente albero delle dipendenze:



Esempio 2.3.14. Nel seguente albero *SimpleNLG* viene mostrato come esempio la frase "*la derivata seconda di f di x rispetto x*".

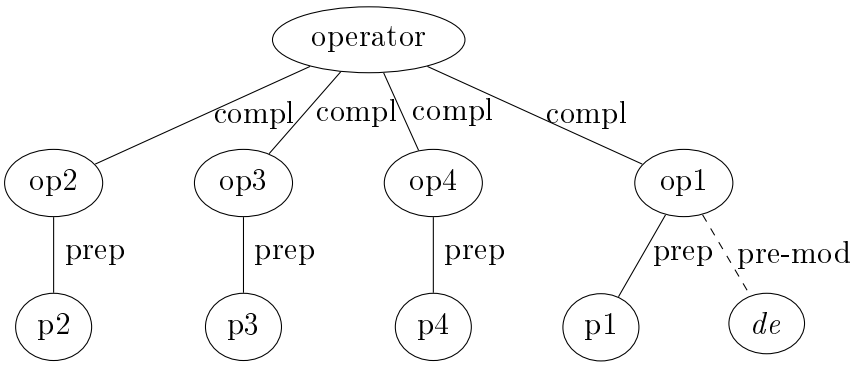


Forma quaternaria

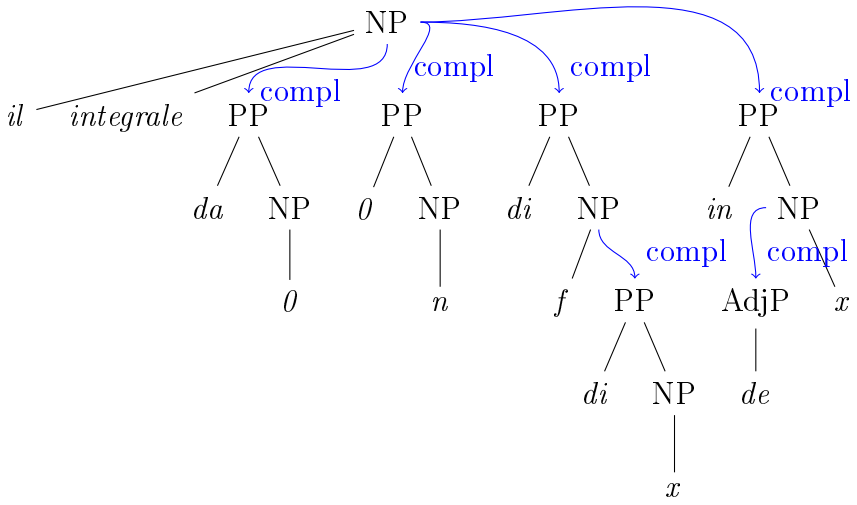
Nel caso in cui l'operatore sia in forma quaternaria (esempio frasi 3) è possibile osservare che le frasi matematiche sono composte da

- un opertore `operator`.
- un operando `op1` rappresentante la variabile legata.
- una preposizione `p1` che lega la variabile legata all'espressione.
- l'eventuale aggettivo "*de*" che precede la variabile legata.
- un operando `op2` rappresentante il limite inferiore.
- una preposizione `p2` che lega l'operatore con il limite inferiore.
- un operando `op3` rappresentante il limite superiore.
- una preposizione `p3` che lega il limite inferiore a quello superiore.
- un operando `op4` rappresentante l'espressione a cui è applicato l'operatore.
- una preposizione `p4` che lega l'operatore all'espressione su cui è applicato.

Le relazioni che intercorrono fra le varie parole possono essere schematizzate con il seguente albero delle dipendenze:



Esempio 2.3.15. Nel seguente albero *SimpleNLG* viene mostrato come esempio la frase "*l'integrale da 0 a n di f di x in de x*".



Capitolo 3

Sintesi delle frasi matematiche

3.1 Differenza tra linguaggio scritto e quello parlato

Il linguaggio parlato e quello scritto, anche se strettamente connessi, non sono del tutto equivalenti. Questo è dovuto al fatto che in entrambi i linguaggi sono presenti degli elementi non appartenenti al lessico che aggiungono significato. Ad esempio nello scritto vi sono la punteggiatura, l'organizzazione in paragrafi e l'uso di maiuscole mentre nel parlato sono presenti le pause, la sillabazione e l'intonazione[1].

3.1.1 Elementi caratterizzanti del parlato

Considerando il linguaggio parlato, elementi tipici che lo caratterizzano sono

- **intonazione**: usata per dare enfasi a determinate parti della frase. Ad esempio in genere si tende ad aumentare il tono alla fine di una frase interrogativa e ad abbassarlo alla fine di una dichiara-

[1] anche la gestualità fa parte del linguaggio parlato ma non verrà considerata nell'ambito di questo lavoro.

tiva. Nello scritto queste informazioni erano invece rappresentate rispettivamente dal punto interrogativo e dal punto.

- **sillabazione**: usata per denotare che una particolare parola è un acronimo. Nello scritto la stessa funzione poteva essere rappresentata dal maiuscolo.

- **pause**: usate per separare sintagmi, frasi e paragrafi. Nello scritto la stessa funzione era svolta dalle virgole e dagli "*a capo*".

3.1.2 SSML

Allo scopo di manipolare gli elementi caratteristici del parlato è stato introdotto un linguaggio noto come `SSML` (*Speech Synthesis Markup Language)*[**?**]. Esso permette di dare delle direttive ai software per la sintesi vocale attraverso dei tag `XML`. Un piccolo esempio di testo arricchito con `SSML` è riportato nel listato seguente.

```
<speak>
    <prosody rate="slow">
        a più <break time="500ms"/> tre diviso due.
    </prosody>
</speak>
```

In questo caso l'attributo *rate* del tag *prosody* indica che la frase deve essere pronunciata lentamente mentre il tag *break* indica che ci deve essere una pausa di 500 ms. Si rimanda alla documentazione ufficiale l'elenco esaustivo dei tag `SSML`.

3.1.3 Matematica parlata

Durante la sintesi delle formule matematiche, l'intonazione non gioca un ruolo importante. Si è infatti lasciato il compito di manipolarla al sintetizzatore vocale, il quale si limita al cambio del tono alla fine della frase.

Seppur in matematica gli acronimi vengono comunemente usati (ad esempio `MCD` per indicare il massimo comune divisore), nelle formule la loro presenza è quasi nulla. Inoltre non sono direttamente rappresentabili dagli operatori disponibili in LaTeX. Per questi motivi durante la fase di sintesi non è stato necessario sfruttare la sillabazione.

Le pause invece ricoprono un ruolo fondamentale nella sintesi delle formule matematiche e il loro utilizzo sarà discusso in modo approfondito nella sezione 3.2.

3.2 Aggregazione delle sottoformule

A livello concettuale un'espressione matematica è rappresentabile da una struttura ad albero chiamata *albero delle espressioni*[2].

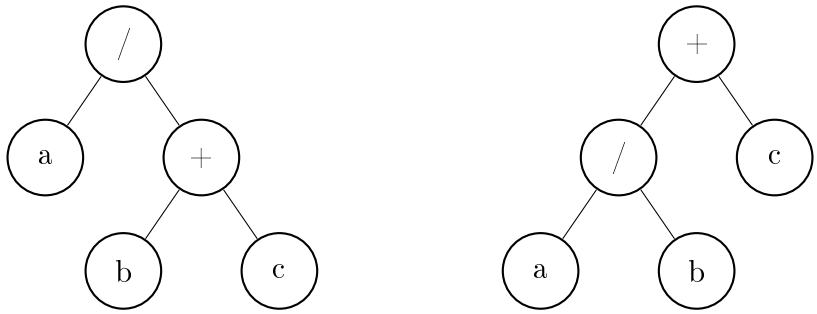


Figura 3.1: La figura 3.1a rappresenta l'espressione $a/(b+c)$ mentre la figura 3.1b rappresenta l'espressione $(a/b)+c$.

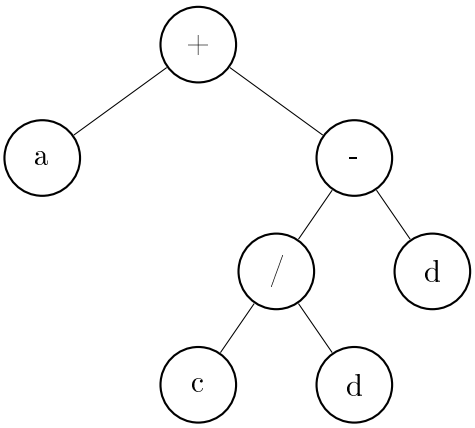
In figura 3.1 si possono vedere due esempi di albero delle espressioni per le formule $a/(b+c)$ e $(a/b)+c$. Come si può osservare questa rappresentazione non presenta alcuna ambiguità in quanto la struttura dell'albero definisce intrinsecamente l'ordine con cui devono essere eseguite le operazioni. Una tale rappresentazione però non si presta bene

[2]questo tipo di albero è molto simile agli alberi di parsing dei linguaggi di programmazione.

quando una formula deve essere comunicata tra esseri umani. Si preferisce infatti una notazione più semplice e compatta. A tale scopo viene eseguita una linearizzazione dell'albero, comprimendolo così in una stringa che può essere usata più agevolmente per trasmettere il significato della formula. Questa operazione di linearizzazione però genera una rappresentazione potenzialmente ambigua. Riconsiderando i due alberi in figura 3.1 si nota che entrambe le linearizzazioni dei due alberi portano ad una stessa rappresentazione:

$$a/b + c$$

È dunque necessario introdurre qualcosa di esterno all'espressione per preservare l'ordine con cui devono essere eseguite le operazioni. La prima cosa che è possibile aggiungere è la parentesizzazione delle sottoformule per specificare come aggregare le sottoformule e quindi preservarne l'ordine di valutazione. Ad esempio la seguente espressione



può essere linearizzata e parentesizzata come segue:

$$(a + ((b/c) - d)) \tag{3.1}$$

Una notazione del genere, seppur corretta, è poco leggibile dato il numero elevato di parentesi. Per diminuirne il numero sono state assegnate delle priorità agli operatori in modo da rimuovere determinate parentesi.

3.2.1 Priorità della matematica scritta

Per convenzione moltiplicazione e divisione hanno un priorità maggiore rispetto alle sottrazioni e divisioni. In questo modo l'espressione 3.1 può essere riscritta come

$$a + b/c - d$$

Questo modalità di parentesizzazione e di assegnamento delle priorità è una convenzione usata nella notazione matematica. Bisogna però considerare che il linguaggio matematico è pensato principalmente per essere scritto. Ci si è quindi chiesti se le stesse convenzioni potessero andare bene anche quando la notazione matematica viene usata nel parlato.

3.2.2 Priorità della matematica parlata

Si è osservato che pronunciando una formula si tende ad assegnare implicitamente delle priorità che non corrispondono con quelle usate nella matematica scritta. Ad esempio nel parlato la formula

"x più uno diviso x meno due"

viene in genere percepita come

$$\frac{x+1}{x-2} \qquad \text{cioè} \qquad (x+1)/(x-2)$$

O ancora

"x più uno per x meno due"

viene in genere percepita come

$$(x+1) * (x-2)$$

Dagli esempi ciò che si evince è che somme e sottrazioni abbiano priorità più alta rispetto alle divisioni e alle moltiplicazioni.

3.2.3 Il ruolo delle pause nella matematica parlata

Si è visto come le priorità degli operatori possano aiutare a diminuire il numero di parentesi. Ovviamente queste ultime sono comunque necessarie per alterare il normale ordine di valutazione delle operazioni.

Nel linguaggio parlato la stessa funzione delle parentesi può essere ottenuta anche con le pause. Si tende infatti ad usarle per indicare l'inizio e la fine delle sottoformule, com'è mostrato nella tabella 3.1.

Parlato	**Scritto**
a più b diviso c meno d	(a + b) / (c - d)
a più <PAUSA> b diviso c meno d <PAUSA>	a + b / (c - d)
a più b diviso c <PAUSA> meno d <PAUSA>	(a + b) / c - d
a più <PAUSA> b diviso c <PAUSA> meno d	a + b / c - d

Tabella 3.1: La tabella mostra alcuni esempi della differenza di parentesizzazione tra formule scritte e parlate.

Le pause sembrano adattarsi meglio al parlato piuttosto che le parentesi. Tuttavia è possibile comunque usare le parentesi per l'aggregazione delle formule. Si vedrà infatti nella sottosezione 3.2.4 che entrambe hanno pregi e difetti.

Notazione 3.2.1. Siccome si è visto che parentesi e pause sono in un certo modo intercambiabili nel parlato, si adotterà il termine "**parentesizzare**" per indicare l'uso di parentesi o pause per aggregare una formula. Si è scelto quindi di usare le parentesi quadre per indicare in modo generale e più compatto l'aggregazione di una formula parlata, indifferentemente dal fatto che venga effettuata con parentesi tonde o pause. Dunque nel parlato la formula

<PAUSA> *a più* <PAUSA> *b diviso c meno d*

e la formula

(*a più*) *b diviso c meno d*

sono entrambi rappresentabili più generalmente da

$$[a \textit{ più }]\ b \textit{ diviso } c \textit{ meno } d$$

3.2.4 Strategie di parentesizzazione

Si è visto che per parentesizzare le formule nel parlato è possibile usare sia le pause che le parentesi. Entrambe le soluzioni mostrano pregi e difetti. Sono quindi state pensate tre diverse strategie di aggregazione:

pause : le pause sembrano una soluzione molto naturale ma se la formula presenta diversi livelli di parentesizzazione diventa difficile comprendere l'inizio e la fine di una sottoformula.

parenthesis: le parentesi permettono di capire con chiarezza l'inizio e la fine di una sottoformula ma sono particolarmente lunghe da pronunciare.

smart : questo metodo è un approccio ibrido e cerca di combinare i lati positivi degli altri due metodi. Consiste nell'usare le pause al livello più interno della parentesizzazione e le parentesi nei restanti livelli in modo da avere una lettura abbastanza veloce e allo stesso comprensibile.

3.2.5 Estensione dell'aggregazione al resto degli operatori

Fino ad ora sono state prese in considerazione soltanto le 4 operazioni, dato che sono gli operatori che generano più ambiguità. Per estendere il metodo di aggregazione a tutti gli altri operatori bisogna considerare che:

- determinati operatori non prevedono alcun tipo di parentesizzazione. Ne sono un esempio gli operatori relazionali. Sono operatori che mettono in relazione due sottoformule; non vi è alcuna possibilità di ambiguità e quindi non necessitano mai di parentesizzazione.

- numeri e variabili non sono ambigui. Anch'essi non necessitano di parentesizzazione.

- tutti gli operatori prefissi agiscono su un'unica espressione. Tecnicamente non tutti gli operatori sono unari. Ad esempio la radice prende due operandi: il grado e l'espressione sotto di essa; la sommatoria prende tre operandi: il limite inferiore, quello superiore e l'espressione che ricade sotto di essa; etc. Ai fini dell'aggregazione però l'unico parametro che ha rilevanza è appunto l'espressione su cui agisce l'operatore, il quale è sempre unico.

Sotto queste considerazioni il metodo di aggregazione si semplifica e si riduce a due casi:

- se l'espressione è un valore semplice, come un numero, una variabile o un altro operatore prefisso allora non serve parentesizzare in quanto non c'è ambiguità su cosa ricada all'interno dell'operatore e cosa no.

- se l'espressione è composita, ovvero è formata da operatori infissi, allora si procede a parentesizzare l'espressione.

Esempio 3.2.1. Considerando i seguenti esempi:

- "*radice quadrata di x*": la radice non necessita di parentesizzazione in quanto il suo argomento è una variabile.

- "*radice quadrata di <PAUSA> x + 1 <PAUSA>*": la radice necessita di parentesizzazione in quanto il suo argomento è una somma.

- "*limite per x tendente a più infinito di radice quadrata di <PAUSA> x + 1 <PAUSA>*": il limite non necessita di parentesizzazione in quanto la radice è a sua volta un operatore infisso. La radice invece necessita di parentesizzazione in quanto il suo argomento è un'espressione composita.

3.3 Algoritmo di Aggregazione

L'idea su cui si basa per l'algoritmo di aggregazione è che durante la trasformazione da rappresentazione semantica a quella sintattica di una formula, si sfruttino implicitamente delle priorità degli operatori per stabilire se parentesizzare o meno una sotto formula.

Verrà presentata inizialmente la versione dell'algoritmo per le formule scritte, in quanto è la notazione con cui si ha più familiarità, per poi mostrare come modificarla per ottenere l'algoritmo per le formule parlate.

3.3.1 Precedenza degli operatori

Si è visto che nelle espressioni scritte, durante la valutazione, si considera che le moltiplicazioni e divisioni abbiano una priorità superiore a quelle delle sottrazioni e addizioni. Ciò lascia intendere che moltiplicazioni e divisioni abbiano la stessa priorità, così come le addizioni e sottrazioni. In realtà si è osservato che nella fase di generazione della rappresentazione sintattica di una formula, le divisioni e sottrazioni hanno rispettivamente una priorità maggiore delle moltiplicazioni e addizioni. Si crea così una gerarchia di priorità mostrata in figura 3.2.

Operatore	**Priorità**
/	4
*	3
-	2
+	1

Figura 3.2: Tabella delle priorità degli operatori matematici

Intuitivamente la motivazione per cui si assegna alla divisione una priorità più alta rispetto alla moltiplicazione, e in modo del tutto analogo anche alle sottrazioni e addizioni, è dovuta principalmente alle proprietà di queste operazioni. La divisione, a differenza della moltiplicazione, non gode della proprietà associativa. Combinando questi due operatori si può vedere che

$$a * b/c = (a * b)/c = a * (b/c)$$

e che quindi il risultato dell'espressione è corretto indifferentemente dall'ordine con cui eseguono le operazioni. Nel caso invece di

$$a/b * c = (a/b) * c \neq a/(b * c)$$

si ottiene una valutazione corretta solo se si esegue la divisione prima della moltiplicazione. Risulta quindi che, anche se in alcuni casi sarebbe possibile valutare le operazioni in qualsiasi ordine, nel caso generale eseguire prima le divisioni delle moltiplicazioni porta sempre a un risultato corretto.

Nella fase di generazione queste priorità possono quindi essere considerate come la precedenza che un operatore ha per accedere agli operandi adiacenti. In altre parole considerando come esempio

$$a * b/c * d$$

si può affermare che la divisione ha la precedenza per accedere a b e c rispetto le altre due moltiplicazioni.

3.3.2 Algoritmo di aggregazione

Per descrivere il funzionamento dell'algoritmo di aggregazione si assume che il suo input sia un albero delle espressioni[3] etichettato con le priorità (si veda figura 3.3).

L'idea è quella di visitare l'albero e per ogni nodo considerare la sua priorità e quella dei suoi figli. Se il nodo genitore ha una priorità più alta rispetto a quella di un nodo figlio allora significa che il nodo genitore ha la precedenza ad accedere all'operando che ha in comune con quel determinato figlio. Ciò però vorrebbe dire andare contro all'ordinamento delle operazioni definite dalla struttura dell'albero. Difatti un operatore che si trova a un livello più basso rispetto ad un altro dovrà per forza avere una priorità più alta sui suoi operandi. In questo caso si procede

[3] Tecnicamente non si usa direttamente questo albero ma ai fini della spiegazione è del tutto ininfluente.

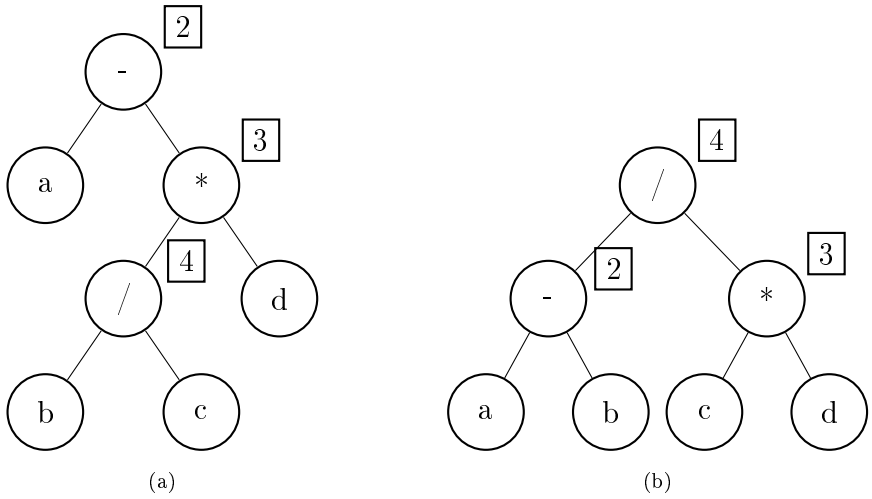


Figura 3.3: La figura 3.1a rappresenta l'espressione $a - b/c * d$ mentre la figura 3.1b rappresenta l'espressione $(a - b)/(c * d)$. I piccioli rettangoli in alto a destra dei nodi operatore indicano la loro priorità

a parentesizzare tutta l'espressione che verrà generata dal nodo figlio in modo da assicurargli l'accesso ai sui operandi prima del nodo genitore. Se invece la priorità del nodo genitore è più bassa rispetto a quello del nodo figlio allora si sta rispettando il normale ordinamento imposto dall'albero e in questo caso la parentesizzazione non è necessaria.

Esempio 3.3.1. Si considerino i due alberi annotati riportati in figura 3.3. Si può osservare come tutti gli operatori dell'albero 3.3a hanno una priorità maggiore rispetto quella dei loro propri figli e che quindi l'espressione generata non necessità di parentesizzazione.

Si può invece osservare che l'operatore di divisione dell'albero 3.3b ha una priorità maggiore rispetto entrambi i suoi figli. È dunque necessario parentesizzare sia l'espressione generata dal nodo della moltiplicazione che quello della sottrazione.

Fino a questo momento si è illustrato il metodo di aggregazione delle formule scritte composte esclusivamente dalla 4 operazioni.

Per poter applicare lo stesso algoritmo alla matematica parlata bisogna considerare che, come indicato alla sottosezione 3.2.2, la priorità degli operatori nel parlato sono inversi rispetto a quelli della scritto. Per questo motivo è possibile o invertire le priorità assegnate agli operatori oppure considerare la priorità più alta tanto quanto si avvicina allo zero. Si è optato per il secondo metodo in quanto prevede solamente di invertire il controllo per la parentesizzazione (da maggiore a minore).

Per estendere invece l'algoritmo a tutti gli altri operatori bisogna considerare che:

- tutti ciò che non prevede parentesizzazione non ha priorità.
- tutti gli operatori prefissi hanno priorità 0, la più alta del parlato.

Algoritmo 1: aggregazione

input : *math-phrase*, *salience-parent*, *salience-children*, *aggregation-method*
output: *math-phrase*

```
1 if salience-parent is not null and salience-children is not null
    and salience-parent < salience-children then
2 |   if aggregation-method = parenthesis then
3 |   |   return math-phrase wrapped between parenthesis
4 |   else
5 |   |   return math-phrase wrapped between pauses
6 |   end
7 else
8 |   return math-phrase
9 end
```

Lo pseudocodice dell'algoritmo è mostrato nel listato 1. Questo algoritmo viene applicato durante la visita dell'albero a tutti i nodi figli del nodo correntemente vistato. Viene preso in input il sintagma matematico generato dal nodo figlio, la precedenza del nodo genitore, quella del figlio

e il metodo di aggregazione che può essere *pause* oppure *parenthesis*. Restituisce se necessario il sintagma matematico parentesizzato in accordo con il metodo di aggregazione o altrimenti lo stesso che ha preso in input.

L'algoritmo gestisce i due metodi di aggregazione *pause* e *parenthesis*. Il metodo *smart* viene invece realizzato usando il metodo *parenthesis* e applicando, tramite un approccio basato su espressioni regolari, la sostituzione delle parentesi più interne con delle pause.

Si può notare come i controlli sulla priorità nulla permettano di evitare di parentesizzare tutti gli elementi che non ne hanno una associata. Inoltre grazie al fatto che la priorità degli operatori infissi è la maggiore di tutte, ovvero 0, è garantito che vengano parentesizzate soltanto le espressioni composite. Infine, dato che il controllo sulla priorità è effettuato con il *minore stretto*, in caso di più operatori infissi annidati solamente quello più interno perentesizzerà se necessario l'espressione su cui è applicato.

3.4 Esempio di aggregazione

Si consideri come esempio albero delle espressioni annotato con le priorità della formula $1 - ((a/2) + b)$: Sintetizzando l'albero e applicando

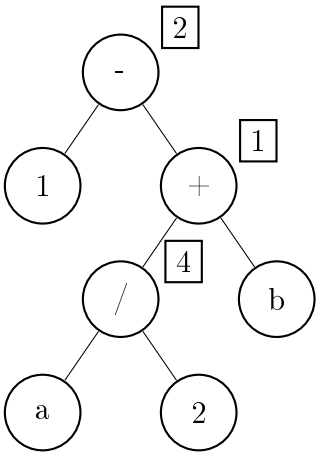


Figura 3.4: L'albero delle espressioni annotato con le priorità della formula $1-((a/2)+b)$.

l'algoritmo di aggregazione nelle tre forme viste si ottiene:

1. **pause**: "*1 meno* <PAUSA><PAUSA> *a diviso due* <PAUSA> *più b* <PAUSA> ".
2. **parenthesis**: "*1 meno* ((*a diviso due*) *più b*) ".
3. **smart**: "*1 meno* (<PAUSA> *a diviso due* <PAUSA>*più b*) ".

Capitolo 4

Design e Implementazione

Come già introdotto nel capitolo 1, la sintesi delle formule matematiche è suddivisa in due fasi principali: quella di analisi e quella di generazione.

4.1 Fase di analisi del LaTeX

Lo scopo di questa fase è quello di ottenere, partendo da una formula LaTeX, una sua rappresentazione in *Content MathML*. Si procederà a dare una breve descrizione del *Content MathML* per poi illustrarne la generazione.

4.1.1 Content MathML

Il *Content MathML* è una rappresentazione semantica espressa mediante il linguaggio `XML` in cui i tag rappresentano operatori matematici o entità di base, come numeri, identificatori, gradi, variabili legate, etc. Ad esempio i numeri e gli indicatori sono espressi da tag unari, come mostrato nel seguente listato

```
1     <cn>4</cn>
2     <ci>a</ci>
```

In *Content MathML* tutti gli operatori matematici sono espressi in forma prefissa. Ad esempio la radice può essere rappresentata in forma prefissa come la funzione binaria *root* che prende in input il grado e l'espressione a cui viene applicata:

$$\sqrt[n]{x} = root(n, x)$$

la quale in *Content MathML* può essere rappresentata come

```
<apply>
    <root/>
    <degree><ci>n</ci></degree>
    <ci>x</ci>
</apply>
```

dove il tag `apply` indica l'applicazione dei parametri ad una funzione.

Un altro esempio significativo è fornito dalla sommatoria. La sua rappresentazione in forma prefissa consiste nella funzione *sum* che prende in input la variabile legata, il limite inferiore, il limite superiore e l'espressione a cui viene applicata:

$$\sum_{i=0}^{n} x = sum(i, 0, n, x)$$

la quale in *Content MathML* viene rappresentata come

```
<apply>
    <sum/>
    <bvar><ci>i</ci></bvar>
    <lowlimit><cn>0</cn></lowlimit>
    <uplimit><cn>0</cn></uplimit>
    <ci>x</ci>
</apply>
```

4.1.2 Produzione del Content MathML

La produzione del *Content MathML* è stata effettuata grazie a uno strumento preesistente chiamato *LatexML* [**?**]. Questo software prende in input una formula LaTeX e restituisce in output una sua rappresentazione

in *Content MathML*. Sfortunatamente il risultato ottenuto non è sempre ciò che ci si aspetterebbe. Difatti in alcuni casi, per via di dell'ambiguità delle formule LaTeX , si ottiene una rappresentazione che non corrisponde con la formula di partenza. Ne è un chiaro esempio l'ambiguità tra l'applicazione e la moltiplicazione. In questo caso l'espressione $f(x)$ viene tradotta come

```
<apply>
    <times/>
    <ci>f</ci>
    <ci>x</ci>
</apply>
```

In questo particolare caso, assumendo che il simbolo f sua usato esclusivamente per indicare una funzione si può semplicemente rimuovere il tag della moltiplicazione ed ottenere il risultato desiderato.

Un altro esempio è quello dell'ambiguità tra coppia e intervallo, cioè $[a, b]$, che si crea quando viene usato una coppia in un insieme condizionale. Difatti in questo contesto l'unico elemento ammesso è la coppia.

In ogni caso il *Content MathML* ottenuto non è direttamente usabile dal sistema in quanto ogni tag è preceduto dal suffisso `m:`, come mostrato nel listato 4.1 . Inoltre tutti gli operatori non definiti in *Content MathML* non vengono rappresentati con dei tag ma indicati all'interno del generico tag `csymbol`. Si preferirebbe però poter lavorare su una struttura omogenea in cui ogni operatore è considerato come un tag.

Si è quindi realizzato uno script che permette di post-processare il *Content MathML* ottenuto da *LatexML* per renderlo compatibile con il sistema.

Esempio 4.1.1. Si consideri l'insieme condizionale $A = \{x \mid x > 0\}$. L'output ottenuto da *LatexML* è il seguente

```
<m:apply>
    <m:eq/>
    <m:ci>A</m:ci>
    <m:apply>
        <m:csymbol cd="latexml">conditional-set</m:csymbol>
        <m:ci>x</m:ci>
        <m:apply>
            <m:gt/>
            <m:ci>x</m:ci>
            <m:cn type="integer">0</m:cn>
        </m:apply>
    </m:apply>
</m:apply>
```

Listing 4.1: *Content MathML* prodotto da LatexML

mentre dopo la fase di post-processamento si ottiene

```
<apply>
    <eq/>
    <ci>A</ci>
    <apply>
        <conditional-set/>
        <ci>x</ci>
        <apply>
            <gt/>
            <ci>x</ci>
            <cn type="integer">0</cn>
        </apply>
    </apply>
</apply>
```

Listing 4.2: *Content MathML* trattato dal PostProcessor

4.2 Fase di Generazione della Frase Matematica

Lo scopo di questa fase è quella di generare il parlato delle formule partendo dalla loro rappresentazione in *Content MathML*. Ciò viene effettuato costruendo l'albero linguistico della formula e poi trasformandolo in linguaggio naturale con l'ausilio della libreria *SimpleNLG-IT*.

4.2.1 L'albero linguistico

Nella sezione 2.3 è stato mostrato che ogni categoria di operatori possiede una propria struttura linguistica. All'interno di ogni categoria però ciascun operatore è caratterizzato da determinati elementi lessicali.

Ad esempio considerando l'operatore "$\geq$" si può notare come la sua traduzione nel sintagma "*è maggiore o uguale a*" è costituita da:

- il verbo *essere* che regge l'operatore
- i due aggettivi, *maggiore* e *uguale*, coordinati con la congiunzione *o*
- la preposizione *a* necessaria a legare il sintagma con il resto della frase

Questi elementi lessicali non sono altro che le foglie degli alberi *SimpleNLG* mostrati nella sezione 2.3.

L'idea è dunque quella di collezionare queste e altre informazioni in un dizionario scritto in formato Json da consultare durante la fase di creazione dell'albero linguistico.

Ad esempio le informazioni più rilevanti dell'operatore "$\geq$" sono rappresentati dalle seguenti proprietà Json:

```
1     "geq": {
2         "name": "maggiore",
3         "coordination-conj": "o",
4         "coordinated-name": "uguale",
5         "preposition": "a",
```

```
6           "verb": "essere",
7       }
```

Proprietà degli Operatori

Non tutti gli operatori necessitano le stesse proprietà. Si procederà quindi a darne una descrizione dettagliata:

name: indica il nome principale dell'operatore. Ad esempio "*valore*" in "*valore assoluto di*" o "*maggiore*" in "*è maggiore o uguale a*".

coordinated-name: indica un eventuale nome coordinato al nome principale come ad esempio "*uguale*" in "*è maggiore o uguale a*".

coordination-conj: indica quale congiunzione usare in caso vi sia una coordinazione di nomi. Ad esempio "*o*" in "*è maggiore o uguale a*".

preposition: indica quale preposizione usare per legare il sintagma corrente con il resto della frase. Ad esempio "*a*" in "*è maggiore o uguale a*" oppure "*di*" in "*maggiore di*".

preposition-alt: indica una preposizione alternativa a quella normale. Viene usata in rari casi quando l'operatore, in un determinato contesto, richiede una preposizione diversa da quella di default. Ad esempio l'operatore di limite inferiore viene in genere usato per indicare il valore da cui parte una sommatoria, una produttoria o un integrale e richiede la preposizione "*da*", come ad esempio nella frase "*sommatoria per x* ***da*** *zero a ...* ". Lo stesso operatore però viene usato anche per indicare il valore a cui tende un limite e in questo particolare caso la proposizione richiesta è "*a*" come ad esempio nella frase "*limite per x tendente* ***a*** *zero*".

preposition-sub: indica quale preposizione usare per legare il sintagma con un eventuale frase subordinata. Ad esempio in "*Il limite per x tendente a più infinito di f di x*", la frase subordinata risulta essere

"*x tendente a più infinito*" e la proposizione che la lega alla frase principale è "*per*".

post-modifier: indica l'eventuale post modificatore del nome. Ad esempio "*assoluto*" in "*valore assoluto di*".

verb: indica quale verbo regge l'operatore. Ad esempio "*essere*" in "*è maggiore o uguale a*" oppure "*tendere*" in "*il limite per x tendente a più infinito di f di x*".

determiner: indica se l'operatore necessita di articolo oppure no. Ad esempio la sommatoria può essere letta in modo più naturale preceduta dell'articolo, mentre l'operatore *più* invece no. È comunque solo un valore di default e può essere ignorato all'occorrenza.

is-negate: specifica se l'operatore è negato oppure no. Ad esempio nella frase "*x non appartiene all'insieme A*" questo valore è impostato a *true*.

is-plural: specifica se l'operatore è plurale oppure no. Ad esempio nella frase "*l'insieme degli x tali che x è maggiore di zero*" questo valore è impostato a *true*.

is-past-participle: specifica se il verbo della frase matematica associata all'operatore è al participio passato. Ad esempio nella frase "*a elevato a 2*" questo valore è impostato a *true*. Se questo valore è impostato a *false* si assume che il verbo sia al presente. Fanno eccezione i verbi usati nei limiti inferiori, i quali sono sempre impostati al participio presente ("*x tendente a ...*", "*x appartenente a ...*").

category: indica la categoria a cui appartiene l'operatore.

Costruzione dell'albero linguistico

A questo punto si è a conoscenza della struttura linguistica di ogni categoria di operatori e degli elementi lessicali associati a ogni operatore. Ciò

che rimane è la costruzione dell'albero nella sua interezza. Per fare ciò si trasforma il *Content MathML* in una struttura ad albero per agevolare la sua navigazione. Dopodiché partendo dalla radice si genera l'albero linguistico considerando l'operatore e i suoi operandi. Nel caso un operando sia a sua volta un operatore si applica ricorsivamente il procedimento su di esso fino a giungere alle foglie dell'albero.

4.2.2 Realizzazione del parlato

La trasformazione dell'albero linguistico in linguaggio naturale si effettua grazie al modulo di realizzazione della libreria *SimpleNLG-IT*. Questo modulo per generare una frase svolge diversi compiti:

- esegue un ordinamento dei i vari sintagmi dell'albero in base al loro legame di dipendenza. Ad esempio in italiano si fa precedere il soggetto al verbo e il verbo all'oggetto.
- coniuga i verbi al tempo specificato. Se nessuno è specificato assume che il tempo sia il presente.
- effettua la flessione degli aggettivi in genere e numero in accordo con quello del nome a cui sono associati.
- effettua la flessione degli nomi in genere e numero se viene esplicitamente richiesto di farlo.
- sceglie la corretta versione degli articoli in base al nome a cui sono associati.
- nega la frase nel caso venga esplicitamente richiesto di farlo.

Fra i compiti appena elencati, quelli che prevedono variazioni morfologiche necessitano di un particolare dizionario che specifica il lessico usato. In caso di lessemi regolari, la libreria è in grado di effettuare queste variazioni in autonomia. Ad esempio se si specifica un verbo regolare, la libreria è in

grado di eseguire la coniugazione. Se invece i lessemi prevedono qualche forma irregolare devono essere specificati manualmente.

Nel contesto delle formule matematiche è stato necessario estendere il dizionario italiano usato di base di *SimpleNLG-IT* con un altro dizionario contenente le parole matematiche, come ad esempio "(*integrale*, N)", "(*derivata*, N)" ...

4.2.3 Esempio di sintesi di una formula

Si prenda nuovamente in considerazione la formula dell'esempio 3.4

$$1 - ((a/2) + b)$$

Per effettuare la sintesi della formula per prima cosa viene data in input al tool *LatexML* e il *Content MathML* ottenuto è il seguente:

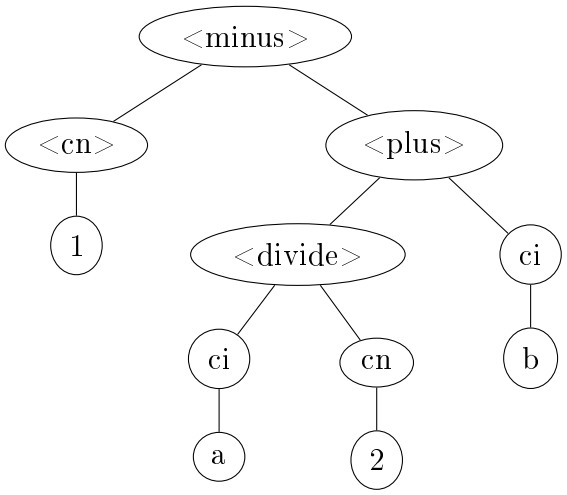
```
<m:apply>
    <m:minus/>
    <m:cn type="integer">1</m:cn>
    <m:apply>
        <m:plus/>
        <m:apply>
            <m:divide/>
            <m:ci>a</m:ci>
            <m:cn type="integer">2</m:cn>
        </m:apply>
        <m:ci>b</m:ci>
    </m:apply>
</m:apply>
```

Dopodiché su usa il modulo di post processamento per trattare il *Content MathML* ottenuto. Il risultato ottenuto è il seguente:

```
<apply>
    <minus/>
    <cn type="integer">1</cn>
    <apply>
        <plus/>
        <apply>
```

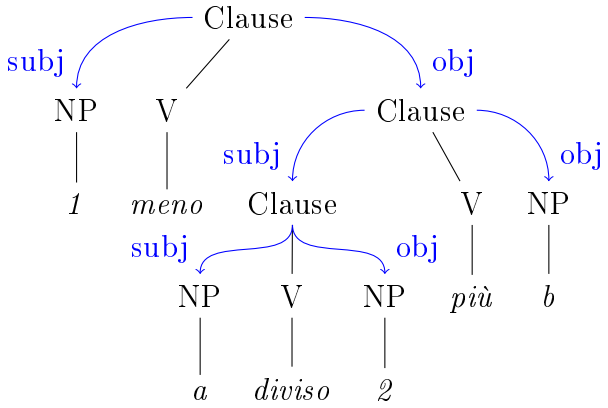
```
7                <divide/>
8                <ci>a</ci>
9                <cn type="integer">2</cn>
10           </apply>
11           <ci>b</ci>
12       </apply>
13   </apply>
```

A questo punto il *Content MathML* viene trasformato in una struttura ad albero nel linguaggio di programmazione clojure. Questa rappresentazione è però molto verbosa e poco leggibile. Inoltre è del tutto equivalente come struttura al *Content MathML*[1]. Si preferisce quindi usare una rappresentazione più semplice da leggere, ovvero la seguente rappresentazione ad albero del *Content MathML*:



Una volta ottenuta questa struttura ad albero si procede a generare l'albero linguistico creando ricorsivamente la struttura linguistica associata ad ogni operatore. L'albero ottenuto è il seguente:

[1] Il *Content MathML* in quanto `XML` è a tutti gli effetti un albero. Può essere visto come un albero che cresce verso destra. Ogni livello di indentazione dei tag rappresenta un livello di profondità.



Infine sfruttando l'algoritmo di aggregazione, ad esempio con la strategia parenthesis, e il modulo del realizer di *SimpleNLG-IT* si ottiene la frase matematica "*1 -((a/2) + b)*".

Capitolo 5

Sperimentazione

Per verificare l'efficacia del sistema è stato preparato un questionario per raccogliere delle valutazioni sulla comprensibilità delle formule sintetizzate. Il sistema è stato progettato pensando ad utenti vedenti e non vedenti ed è stato realizzato mediante il tool *Google Form*[1]. Il questionario prevede una sezione iniziale di profilazione, seguita da 25 sezioni contenenti:

- un audio della sintesi vocale di una formula matematica generata dal sistema
- una domanda a risposta aperta in cui si richiede di scrivere la formula ascoltata usando una notazione non ambigua
- una domanda a risposta multipla in cui si chiede di valutare la facilità di comprensione della formula su una scala di Likert a 7 risposte

Le formule fatte ascoltare ai soggetti del test sono suddivise in due categorie: formule facili e formule difficili. I criteri usati per decidere a quale categoria appartiene una formula sono 2:

[1]Il sondaggio è reperibile al seguente link: `https://docs.google.com/forms/d/1PMl_bKVkxDpzGHQeEj4bG_CPK73mIyZvjvbHzVRjglE/edit`

- il numero di nodi dell'albero generato dal *Content MathML*.

- la necessità di parentesizzazione

Combinando questi due criteri si è stabilito che tutte le formule che non necessitano parentesizzazione e che abbiano un numero di nodi inferiore a 15 siano classificate come facili, si veda tabella 5.1. Il numero 15 è il valore di riferimento che corrisponde al numero di nodi della definizione del prodotto cartesiano.

Le formule che invece hanno un alto numero di nodi e necessitano di parentesizzazione vengono classificate come difficili, si veda tabella 5.2. Si può notare nella tabella delle formule difficili che la definizione di numero di Nepero ha solamente 10 nodi. Questa formula è però stata considerata difficile perché prevede due livelli annidati di parentesizzazione.

Formula LaTeX	**Nodi**
$A \times B = \{(x, y) \mid x \in A, y \in B\}$	15
$g^{-1}(y) = f^{-1}\left((y-b)/a\right)$	13
$\int_b^c a \, \mathrm{d}x = a(c-b)$	14
$x > b \Longrightarrow \lvert f(x)\rvert < M$	10
$\sqrt[n]{x} = x^{1/n}$	10

Tabella 5.1: Le formule usate durante il test e considerate facili.

Formula LaTeX	Nodi
$\lim_{x \to x_0} \left\{ \frac{f(x) - f(x_0)}{x - x_0} - f'(x_0) \right\} = 0$	31
$y = f(a) + \frac{f(b) - f(a)}{b - a}(x - a)$	21
$\int \frac{1}{\sqrt{m^2 - x^2}} \mathrm{d}x = \arcsin \frac{x}{m} + c$	20
$\sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k$	28
$\lim \left(1 + \frac{1}{n}\right)^n = e$	10

Tabella 5.2: Le formule usate durante il test e considerate difficili.

5.1 Risultati

La fase di sperimentazione è tuttora aperta e al momento della scrittura della tesi, 5 persone hanno partecipato alla sperimentazione fra cui 4 non vedenti e 1 vedente. Purtroppo una delle valutazioni non è potuta essere accettata in quanto priva di parentesizzazione delle formule, elemento cruciale ai fini del test. È stato inoltre riportato che il tempo per compilare l'intero questionario è di oltre una ora.

Sono state scelte in totale 10 formule, 5 facili e 5 difficili. Le formule sono state proposte con sintetizzatori vocali diversi e aggregate con i 3 diversi modi. Inoltre a tutte le formule presentate in versioni diverse si sono rinominati i simboli di funzione e parametri. Ad esempio $f(x)$ in un test, $g(y)$ in un altro, etc.

I sintetizzatori usati sono quello fornito da `IBM` [?] e *espeak* [?]. Le strategie sono invece le tre viste alla sottosezione 3.2.4: *pause*, *par* e *smart*.

Per ogni formula si riporterà la percentuale di persone che hanno tra-

scritto la formula in modo corretto. In caso contrario sarà indicato il tipo di errore:

- errore di parentesizzazione: se è stata percepita una parentesizzazione della formula diversa da quella attesa. Verrà indicato con la lettera *P*.
- errore simbolico: se l'errore riguarda la comprensione dei simboli pronunciati. Verrà indicata con la lettera *S*.

5.1.1 Formule facili

Formula LaTeX	**Aggr.**	**Sint.**	**Corr. %**	**Tipo Err.**
$A \times B = \{(x, y) \mid x \in A, y \in B\}$	smart	ibm	100	-
$g^{-1}(y) = f^{-1}\left((y-b)/a\right)$	smart	ibm	25	P
$\int_b^c d \, \mathrm{d}x = d(c-b)$	smart	ibm	25	P
$x > b \Longrightarrow \lvert f(x)\rvert < M$	smart	ibm	100	-
$\sqrt[n]{x} = x^{1/n}$	pause	ibm	75	S
$C \times D = \{(z, y) \mid z \in C, y \in D\}$	smart	espeak	100	-
$z > k \Longrightarrow \lvert g(z)\rvert < C$	pause	espeak	100	-

Tabella 5.3: I risultati ottenuti con le formule facili. La colonna Aggr indica la strategia di aggregazione, Sint indica il tipo di sintetizzatore usato, Corr % indica la percentuale di persone che ha compreso correttamente la formula e Tipo Err indica il tipo di errore commesso: S errore simbolico, P errore di parentesizzazione.

Osservazioni

- Il 75% dei soggetti ha percepito la seconda formula come

$$g^{-1}(y) = f^{-1}\left((y)\right) - b/a$$

- Il 75% dei soggetti ha percepito la terza formula come

$$\int_b^c d \, \mathrm{d}x = (d * c) - b$$

- Il 25% dei soggetti ha percepito la quinta formula come

$$\sqrt{x} = x^{1/n}$$

5.1.2 Formule difficili

Formula LaTeX	**Aggr.**	**Sint.**	**Corr. %**	**Tipo Err.**
$\lim_{x \to x_0} \left\{ \frac{f(x) - f(x_0)}{x - x_0} - f'(x_0) \right\} = 0$	pause	ibm	75	P
$y = f(c) + \frac{f(d) - f(c)}{d - c}(x - c)$	pause	ibm	25	S, P
$\int \frac{1}{\sqrt{k^2 - x^2}} \mathrm{d}x = \arcsin \frac{x}{k} + c$	pause	ibm	25	P
$\sum_{k=0}^{n} \frac{g^{(k)}(x_0)}{k!}(x - x_0)^k$	pause	ibm	100	-
$\lim \left(1 + \frac{1}{n}\right)^n = e$	pause	ibm	75	S

Tabella 5.4: I risultati ottenuti dal primo blocco di formule difficili. La colonna Aggr indica la strategia di aggregazione, Sint indica il tipo di sintetizzatore usato, Corr % indica la percentuale di persone che ha compreso correttamente la formula e Tipo Err indica il tipo di errore commesso: S errore simbolico, P errore di parentesizzazione.

Osservazioni

- Il 25% dei soggetti ha percepito la prima formula come

$$\lim_{x \to x_0} \left\{ f(x) - \frac{f(x_0)}{x - x_0} - f'(x_0) \right\} = 0$$

- Il 25% dei soggetti ha percepito la seconda formula come

$$y = f(c) + a\frac{f(d) - f(c)}{d - c}(x - c)$$

  mentre il 50% l'ha percepita come

$$y = f(c) + \frac{f(d) - f(c)}{d - c}x - c$$

- Il 25% dei soggetti ha percepito la terza formula come

$$\int \frac{1}{\sqrt{k^2}} - x^2 \mathrm{d}x = \arcsin \frac{x}{k} + c$$

  mentre il 25% come

$$\int \frac{1}{\sqrt{k^2 - x^2}} \mathrm{d}x = \frac{\arcsin(x)}{k} + c$$

  e il restante 25% come

$$\int \frac{1}{\sqrt{k^2 - x^2}} \mathrm{d}x = \arcsin(\frac{x}{k} + c)$$

- Il 25% dei soggetti non ha percepito la quinta formula come

$$\left(\lim 1 + \frac{1}{n} \right)^n = e$$

Formula LaTeX	Aggr.	Sint.	Corr. %	Tipo Err.
$\lim_{z \to z_0} \left\{ \frac{g(z) - g(z_0)}{z - z_0} - g'(z_0) \right\} = 0$	par	ibm	25	S,P
$y = g(b) + \frac{g(c) - g(b)}{c - b}(z - b)$	par	ibm	0	S
$\int \frac{1}{\sqrt{s^2 - y^2}} \mathrm{d}y = \arcsin \frac{y}{s} + c$	par	ibm	50	P
$\sum_{n=0}^{l} \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n$	par	ibm	100	-
$\lim \left(1 + \frac{1}{x}\right)^x = e$	par	ibm	100	-

Tabella 5.5: I risultati ottenuti dal secondo blocco di formule difficili. La colonna Aggr indica la strategia di aggregazione, Sint indica il tipo di sintetizzatore usato, Corr % indica la percentuale di persone che ha compreso correttamente la formula e Tipo Err indica il tipo di errore commesso: S errore simbolico, P errore di parentesizzazione.

Osservazioni

- Il 25% dei soggetti ha percepito la prima formula come

$$\lim_{x \to x_0} \left\{ f(x) - \frac{f(x_0)}{x} - x_0 - f'(x_0) \right\} = 0$$

  mentre il 50% non è riuscita a capire e scrivere buona parte delle formula.

- il 75% ha riportato la variabile a che non appartiene alla seconda formula e il restante 25% ha aperto una parentesi tonda senza chiuderla rendendo impossibile valutare la correttezza della formula.

- Il 25% dei soggetti ha percepito la terza formula come

$$\int \frac{1}{\sqrt{s^2 - y^2}} \mathrm{d}y = \arcsin(y)/s + c$$

mentre l'altro 25% come

$$\int \frac{1}{\sqrt{s^2 - y^2}} \mathrm{d}y = \arcsin y/s + c$$

Formula LaTeX	**Aggr.**	**Sint.**	**Corr. %**	**Tipo Err.**
$\lim_{y \to y_0} \left\{ \frac{h(y) - h(y_0)}{y - y_0} - h'(y_0) \right\} = 0$	smart	ibm	0	S
$y = h(s) + \frac{h(t) - h(s)}{t - s}(z - s)$	smart	ibm	0	P
$\int \frac{1}{\sqrt{l^2 - x^2}} \mathrm{d}x = \arcsin \frac{x}{l} + c$	smart	ibm	50	P
$\sum_{k=0}^{n} \frac{h^{(k)}(y_0)}{k!}(y - y_0)^k$	smart	ibm	100	-
$\lim \left(1 + \frac{1}{k}\right)^k = e$	smart	ibm	100	-

Tabella 5.6: I risultati ottenuti dal terzo blocco di formule difficili. La colonna Aggr indica la strategia di aggregazione, Sint indica il tipo di sintetizzatore usato, Corr % indica la percentuale di persone che ha compreso correttamente la formula e Tipo Err indica il tipo di errore commesso: S errore simbolico, P errore di parentesizzazione.

Osservazioni

- Il 100% ha sbagliato a comprendere la prima formula confondendo "*La derivata di h*" con "*La derivata h-esima*".
- Il 100% ha percepito erroneamente le parentesi della frazione della seconda formula.
- Il 25% ha percepito la terza formula come

$$\int \frac{1}{\sqrt{l^2 - x^2}} \mathrm{d}x = \arcsin(x)/l + c$$

mentre l'altro 25% come

$$\int \frac{1}{\sqrt{l^2 - x^2}} \mathrm{d}x = \arcsin(x/l + c)$$

Formula LaTeX	**Aggr.**	**Sint.**	**Corr. %**	**Tipo Err.**
$\lim_{x \to x_0} \left\{ \frac{h(x) - h(x_0)}{x - x_0} - h'(x_0) \right\} = 0$	pause	espeak	0	P
$\int \frac{1}{\sqrt{t^2 - z^2}} \mathrm{d}z = \arcsin \frac{z}{t} + c$	pause	espeak	50	P
$\lim \left(1 + \frac{1}{t}\right)^t = e$	pause	espeak	100	-

Tabella 5.7: I risultati ottenuti dal quarto blocco di formule difficili. La colonna Aggr indica la strategia di aggregazione, Sint indica il tipo di sintetizzatore usato, Corr % indica la percentuale di persone che ha compreso correttamente la formula e Tipo Err indica il tipo di errore commesso: S errore simbolico, P errore di parentesizzazione.

Osservazioni

- Il 100% ha percepito erroneamente le parentesi della frazione della prima formula.

- Il 25% ha percepito la terza formula come

$$\int \frac{1}{\sqrt{t^2 - z^2}} \mathrm{d}z = \arcsin(z)/t + c$$

  mentre l'altro 25% come

$$\int \frac{1}{\sqrt{t^2 - z^2}} \mathrm{d}z = \arcsin(z/t + c)$$

5.2 Analisi dei risultati

Dai risultati sperimentali ottenuti si distingue nettamente che il sistema si comporta in modo migliore con formule più semplici. La colonna *% Corr.* corrisponde alla percentuale di soggetti che sono riusciti a comprendere correttamente le formule nella sua interezza. Inoltre dato che in diversi casi la formula non è stata compresa in quanto non si sono riusciti a capire quali simboli sono stati pronunciati dal sintetizzatore (errore di tipo S), si è deciso di riportare anche le percentuali che tengono in considerazione solo gli errori di parentesizzazione. In questo modo è possibile dare una valutazione più accurata sui metodi di parentesizzazione.

Bisogna comunque considerare che il campione dei soggetti che hanno partecipato al test è molto ristretto, si tratta di 4 persone, e che quindi i risultati sono molto sensibili alle piccole variazioni.

I risultati inerenti alle formule semplici sono riportati in tabella 5.8. Da notare che le due formule pronunciate con *espeak* sono state com-

Sintetizzatore	**Aggregazione**	**% Totale Corrette**	**% Corrette senza errore S**
ibm	smart	65	70
espeak	smart	100	100

Tabella 5.8: Risultati delle formule facili. % totale corrette indica la percentuale totale di persone che hanno compreso la formula nella sua interezza e % Corrette senza errore S indica quante persone hanno compreso correttamente una formula escludendo gli errori simbolici.

prese dalla stessa percentuale di soggetti anche quando pronunciate dal sintetizzatore di `IBM`.

Per quanto riguarda le formule difficili i risultati sono riportati nella tabella 5.9.

Sintetizzatore	**Aggregazione**	**% Totale Corrette**	**% Corrette senza errore S**
ibm	pause	60	66.6
ibm	par	55	60
ibm	smart	50	62.5
espeak	-	50	50

Tabella 5.9: Risultati delle formule difficili. % totale corrette indica la percentuale totale di persone che hanno compreso la formula nella sua interezza e % Corrette senza errore S indica quante persone hanno compreso correttamente una formula escludendo gli errori simbolici.

Inoltre nei commenti del questionario è stato espressa una preferenza riguardante i due sintetizzatori. Al contrario di ciò che si pensava è stato preferito il sintetizzatore di espeak rispetto quello delle `IBM`.

Capitolo 6

Conclusioni

L'idea alla base di questo lavoro è stata quella di poter fornire a persone con disabilità visive uno strumento che gli permettesse di comprendere in maniera più agevole le formule matematiche presenti nei documenti scientifici. Al momento è stato realizzato un sistema software funzionante che permette di sintetizzare la lettura naturale delle formule matematiche costituite da un inseme di operatori matematici. Anche se questo insieme di operatori non è completo si riescono comunque a rappresentare un numero considerevole delle formule presenti nel Pandolfi.

6.1 Il lavoro svolto

Il lavoro svolto in questa tesi si è articolato principalmente in 3 fasi:

1. **analisi linguistica delle formule matematiche**: in questa si è indagato sulla natura linguistica delle formule matematiche. Si è osservato che anche se non tutte hanno una struttura linguistica compatibile con quella del linguaggio naturale, sotto opportune ipotesi è comunque possibile trattarle come se fossero a tutti gli effetti frasi del linguaggio naturale.

2. **sintesi delle frasi matematiche**: in questa fase si è invece affrontato il problema della sintesi delle formule matematiche. Si è visto che le frasi matematiche scritte e parlate seguono regole diverse. Si è quindi investigato su modi alternativi per l'aggregazione delle formule allo scopo di permettere la loro sintesi in maniera più semplice e si sono presentate tre strategie diverse di parentesizzazione.

3. **sperimentazione**: in questa fase si è chiesto a dei soggetti con disabilità visive di testare il sistema e dare una sua valutazione. Dai risultati raccolti si è visto che la percentuale di persone che è riuscita a comprendere la formula nella sua interezza è circa il 75% nel caso di formule semplici mentre scende intorno al 60% nel caso di formule più complesse. Nella maggior parte dei casi tutti gli operatori e i simboli di funzione e variabili sono stati compresi ma l'aggregazione delle sottoformule è stata percepita in maniera differente da quella che si si attendeva. Uno delle possibili cause di ciò può dipendere dal fatto che ai soggetti non è stato illustrato il ruolo delle pause nell'aggregazione per ottenere dei risultati più naturali. Ciò però potrebbe avere generato confusione fra la parentesizzazione scritta e quella parlata.

6.2 Possibili lavori futuri

Si possono pensare diverse estensioni del lavoro di tesi per migliorare il sistema e facilitarne il suo utilizzo. Fra i più rilevanti vi sono:

- estendere l'insieme degli operatori gestiti dal sistema.
- creare altri dizionari matematici per poter gestire la sintesi anche in altre lingue.
- sviluppare un sistema che permetta di ottenere il *Content MathML* in modo più semplice.

- sviluppare un software che permetta la navigazione della formula. In questo modo l'utente può richiedere la pronuncia anche solo di una parte della formula (sottoformula) che è quindi più facile da comprendere.

- integrare il sistema con altri strumenti, in particolar modo i browser.

Infine si può immaginare di usare nella valutazione dei risultati la tecnica adoptata normalmente nell'ambito del parsing del linguaggio naturale, come ad esempio *Eval B* o *Pareval* (Jurafsky, [**?**]). In questo modo si riuscirebbe ad avere una valutazione più fine degli errori di comprensione della struttura.