

Università degli Studi di Trento
Dipartimento di Economia e Management

elm-interpreter-fork:

modello dati multidimensionale (XModel)
e valutazione vettoriale per elm-interpreter,
motore di calcolo di elm-foldbook

Nota tecnica sul prodotto della ricerca di tipo *Software*

Luca Erzegovesi

Versione	commit bc148ee (18 dicembre 2024)
Periodo di sviluppo	giugno – dicembre 2024; pubblico dal 29 luglio 2024
Repository	https://github.com/lerzegov/elm-interpreter-fork
Progetto originale	https://github.com/miniBill/elm-interpreter (L. Taglialegne)
DOI	10.5281/zenodo.23056240
SWHID	swh:1:rev:bc148ee3cdb33f435fd68f575a508c01444768ea
Usato da	elm-foldbook, DOI 10.5281/zenodo.23056100
Licenza	BSD 3-Clause

Trento, 14 ottobre 2026

Scheda del prodotto

Titolo	elm-interpreter-fork: modello dati multidimensionale (XModel) e valutazione vettoriale per elm-interpreter, motore di calcolo di elm-foldbook
Tipologia	Software
Autore	Luca Erzegovesi, Università di Trento
Sviluppo	giugno – dicembre 2024: dentro il repository di elm-foldbook fino al 29/07/2024, poi nel fork pubblico (10 commit dell'autore, 29/07/2024 – 18/12/2024)
Versione di riferimento	commit bc148ee, 18/12/2024 (ultimo push pubblico)
Linguaggio	Elm 0.19.1
Dimensioni	7.102 righe di codice Elm dell'autore su 11.890 della cartella src/; il resto è l'interprete di Leonardo Taglialegne
Licenza	BSD 3-Clause, la stessa del progetto originale
Stato	motore di calcolo di elm-foldbook 0.7; modello dati ripreso nel front end di Impromptu (2026)
Disponibilità	repository GitHub pubblico; archivio Software Heritage; deposito Zenodo con DOI
Identificativi	DOI 10.5281/zenodo.23056240; SWHID swh:1:rev:bc148ee3cdb33f435fd68f575a508c01444768ea

Sommario

elm-interpreter-fork è il motore di calcolo di elm-foldbook (Erzegovesi 2026a), il prototipo di foglio di calcolo multidimensionale descritto in una nota tecnica distinta. Parte da elm-interpreter di Leonardo Taglialegne (Taglialegne 2023), un interprete del linguaggio Elm scritto in Elm, e vi aggiunge tre cose: un modello dati multidimensionale (XModel), con dataset, dimensioni dotate di nome e array piatti indicizzati per coordinata; un nuovo tipo di valore dell'interprete che rappresenta questi array; l'estensione dell'aritmetica dell'interprete agli array, con allineamento automatico per coordinate. Il risultato è un linguaggio di formula per fogli multidimensionali che coincide con un linguaggio di programmazione completo, senza scrivere un parser nuovo. Il codice è stato sviluppato fra giugno e dicembre 2024 ed è pubblico su GitHub dal 29 luglio 2024. Questa nota descrive il contributo dell'autore al fork; l'applicazione, l'interfaccia e i documenti sono descritti nella nota di elm-foldbook.

Abstract

elm-interpreter-fork is the calculation engine of elm-foldbook, a multidimensional spreadsheet prototype. It extends Leonardo Taglialegne's elm-interpreter, an Elm interpreter written in Elm, with a multidimensional data model (XModel: datasets, named dimensions, flat arrays indexed by coordinate), a new interpreter value that carries these arrays, and array arithmetic with automatic alignment by coordinate. The outcome is a formula language for multidimensional spreadsheets that is a complete programming language, obtained without writing a new parser. Developed between June and December 2024, public on GitHub since 29 July 2024.

Indice

Scheda del prodotto	1
1 Il problema affrontato	3
1.1 Un linguaggio di formula per modelli multidimensionali	3
1.2 Perché un interprete di Elm	3
1.3 Obiettivi	3
2 Stato dell'arte	3
2.1 Array con dimensioni dotate di nome	3
2.2 Linguaggi di formula	3
2.3 Interpreti incorporati	4
2.4 Lo spazio coperto	4
3 Contributo originale	4
3.1 Cosa è originale e cosa è ereditato	4
3.2 I contributi principali	4
3.3 Cronologia dello sviluppo	5
4 Architettura	6
4.1 Visione d'insieme	6
4.2 Il modello dati XModel	6
4.3 Nomi di intervallo	7
4.4 L'innesto nell'interprete	8
4.5 Aritmetica vettoriale e allineamento	8
4.6 Valori delle celle e funzioni finanziarie	9
5 Esempi d'uso	9
5.1 Un modulo di formule	9
5.2 Ricalcolo a catena	10
5.3 Formule fra dataset	10
6 Limiti e sviluppi	10
6.1 Limiti	10
6.2 Che cosa ne è seguito	11
7 Disponibilità, licenza e riproducibilità	11
7.1 Dove si trova	11
7.2 Licenza	11
7.3 Compilare ed eseguire	11
Ringraziamenti	11

Riferimenti bibliografici	12
A Moduli del fork	12

1 Il problema affrontato

1.1 Un linguaggio di formula per modelli multidimensionali

Un foglio di calcolo multidimensionale, nella tradizione di Lotus Improv e di Quantrix Modeler (Quantrix, s.d.), organizza i dati in dimensioni dotate di nome (azienda, anno, voce di bilancio, scenario) e scrive una formula una sola volta per un'intera porzione del modello: *EBITDA = margine di contribuzione – spese generali* vale per tutte le aziende e per tutti gli anni. Serve quindi un linguaggio di formula in cui una variabile denota un *array* con dimensioni dotate di nome e in cui le operazioni aritmetiche si applicano agli array interi, allineandoli per coordinata.

Le strade sono due. La prima è inventare una sintassi apposita, con il suo parser, il suo valutatore e le sue funzioni: è la scelta dei prodotti commerciali e porta a linguaggi chiusi, che l'utente non può estendere. La seconda è riusare un linguaggio di programmazione esistente e insegnargli a trattare gli array del modello. *elm-interpreter-fork* segue la seconda strada.

1.2 Perché un interprete di Elm

elm-foldbook è un'applicazione web scritta in Elm (Czaplicki e Chong 2013) che deve funzionare tutta nel browser, senza un server di calcolo. *elm-interpreter* (Taglialegne 2023) è un interprete di Elm scritto in Elm: analizza un sorgente con la libreria *elm-syntax* (Stil4m, s.d.) e lo valuta nel browser, con una libreria standard (*elm/core*) rigenerata per l'interprete. Usarlo come motore significa avere gratis un linguaggio tipizzato con funzioni, *let*, *case*, liste e funzioni di ordine superiore. Mancava però tutto ciò che serve a un foglio di calcolo: un modello dei dati, un modo di nominare porzioni del modello dentro le formule e un'aritmetica sugli array.

1.3 Obiettivi

1. Rappresentare un modello come insieme di dataset che condividono dimensioni con nome.
2. Permettere alle formule di riferirsi a porzioni del modello con nomi validi in Elm.
3. Estendere gli operatori e le funzioni matematiche dell'interprete agli array, con *broadcasting* fra scalari e array e allineamento fra array di forma diversa.
4. Valutare le formule di un dataset senza rileggerne ogni volta il sorgente.
5. Offrire all'interfaccia le operazioni di selezione e di modifica della struttura (aggiungere anni, voci, scenari) necessarie a una griglia di tipo pivot.

2 Stato dell'arte

2.1 Array con dimensioni dotate di nome

L'idea di operare su array interi risale ad APL (Iverson 1962). Nel calcolo scientifico gli array con dimensioni e coordinate etichettate sono oggi uno standard: *pandas* (McKinney 2010) e *xarray* (Hoyer e Hamman 2017) in Python, *DimensionalData.jl* in Julia. Selezione per coordinata (*loc*) e per posizione (*iloc*), *broadcasting* e allineamento automatico nelle operazioni sono proprio ciò che un modello finanziario multidimensionale richiede. Sono però librerie per programmatori, non linguaggi di formula di un foglio di calcolo.

2.2 Linguaggi di formula

I fogli multidimensionali commerciali usano linguaggi di formula propri, scritti per nome di categoria e di elemento ma non estensibili dall'utente. Excel ha introdotto gli array dinamici e la funzione

LAMBDA (Peyton Jones et al. 2003), che avvicinano il foglio a un linguaggio funzionale; il modello resta però legato agli indirizzi di cella.

2.3 Interpreti incorporati

Incorporare un interprete di un linguaggio general purpose in un'applicazione è una tecnica consolidata (per esempio i linguaggi di scripting dei programmi di calcolo). Nel caso di Elm, `elm-interpreter` è l'unico interprete scritto nello stesso linguaggio e quindi eseguibile dentro un'applicazione Elm. Non risultano all'autore estensioni dell'interprete a valori multidimensionali.

2.4 Lo spazio coperto

Il fork combina le tre linee: un modello di array con dimensioni dotate di nome, innestato in un interprete di un linguaggio funzionale tipizzato, che diventa così il linguaggio di formula di un foglio di calcolo multidimensionale eseguito nel browser.

3 Contributo originale

3.1 Cosa è originale e cosa è ereditato

Il fork è un fork GitHub di `miniBill/elm-interpreter` creato il 29/07/2024. La cronologia contiene 311 commit di Leonardo Taglialegne, 2 di Pit Capitain, 6 automatici e 10 dell'autore. La Tabella 1 distingue il codice dell'autore da quello ereditato, contando le righe della cartella `src/` con `git blame` alla revisione `bc148ee`.

Tabella 1: Righe di codice Elm della cartella `src/` per origine (commit `bc148ee`).

Gruppo di moduli	Righe	di L. Erze- govesi
Modello dati e servizi (<code>XModel</code> , <code>TypesXModel</code> , <code>XModelSample</code> , <code>XParser</code> , <code>FormulaParser</code> , <code>Financial</code> , <code>AppUtil</code> , <code>XModelHelpers</code>)	6.326	6.326
Kernel (primitive dell'interprete)	869	163
<code>Eval/Expression</code> (valutatore)	1.406	201
<code>Eval/Module</code> (valutazione dei moduli)	321	211
<code>Types</code> , <code>Value</code> , <code>Environment</code> e altri moduli di base	1.878	201
Moduli dell'interprete non modificati	1.090	–
Totale <code>src/</code>	11.890	7.102

Sono di Taglialegne, e non modificati in modo sostanziale, il parser (attraverso `elm-syntax`), il valutatore di base, la libreria `elm/core` rigenerata con `elm-codegen` (cartelle `codegen/`, `helpers/`, `generated/`) e i test. Il conteggio delle righe non misura il valore intellettuale: l'interprete di Taglialegne è la base senza la quale il motore non esisterebbe. Il contributo dell'autore sta nel modello dati e nel suo innesto nell'interprete.

3.2 I contributi principali

1. **Il modello dati `XModel`.** Dimensioni condivise, dataset, array piatti con passi (*strides*), selezione per coordinata e per posizione, modifica della struttura con riallineamento dei dati (Sezione 4.2).

2. **Un nuovo valore dell'interprete.** Il costruttore DataAr porta un array del modello dentro l'interprete come un qualsiasi valore Elm; i nomi di intervallo sono risolti in modo pigro durante la valutazione (Sezione 4.4).
3. **Aritmetica vettoriale con allineamento.** Operatori aritmetici e funzioni matematiche accettano scalari e array in ogni combinazione; array di forma diversa sono allineati per coordinata (Sezione 4.5).
4. **Nomi di intervallo e completamento.** Una grammatica che rende ogni porzione del modello un identificatore Elm valido, il suo inverso e la proposta delle coordinate compatibili con un nome scritto a metà (Sezione 4.3).
5. **Funzioni finanziarie** per obbligazioni, valore attuale, tasso interno di rendimento e opzioni (Sezione 4.6).

3.3 Cronologia dello sviluppo

Tabella 2: Cronologia del fork.

Data	Passaggio
19/06/2024	elm-foldbook nasce dalla cronologia di elm-interpreter; i primi tipi del modello sono scritti nel repository dell'applicazione
26/06/2024	elm-foldbook v0.2: nomi di intervallo (rangeNameToDef) e completamento nell'editor
01/07/2024	elm-foldbook v0.3: ricalcolo fra dataset
29/07/2024	Creazione del fork pubblico; aggiunte le funzioni finanziarie; XModel integrato nell'interprete, elm-syntax aggiornato alla 7.3.4 (7 commit); elm-foldbook 0.5 compila il motore dal fork
ago.–dic. 2024	Sviluppo usato da elm-foldbook 0.6 (12/08/2024) e 0.7 (16/12/2024)
18/12/2024	Due commit: revisione di XModel, Kernel ed Eval/Module; modello di esempio spostato in XModelSample (bc148ee, ultimo push)
30/09/2026	Archiviazione in Software Heritage e deposito su Zenodo del commit bc148ee, senza modifiche

Le date del fork sono attestate dai metadati del server GitHub (creazione 29/07/2024, ultimo push 18/12/2024). GitHub non consente fork privati di repository pubblici, quindi il fork è pubblico dalla creazione.

4 Architettura

4.1 Visione d'insieme

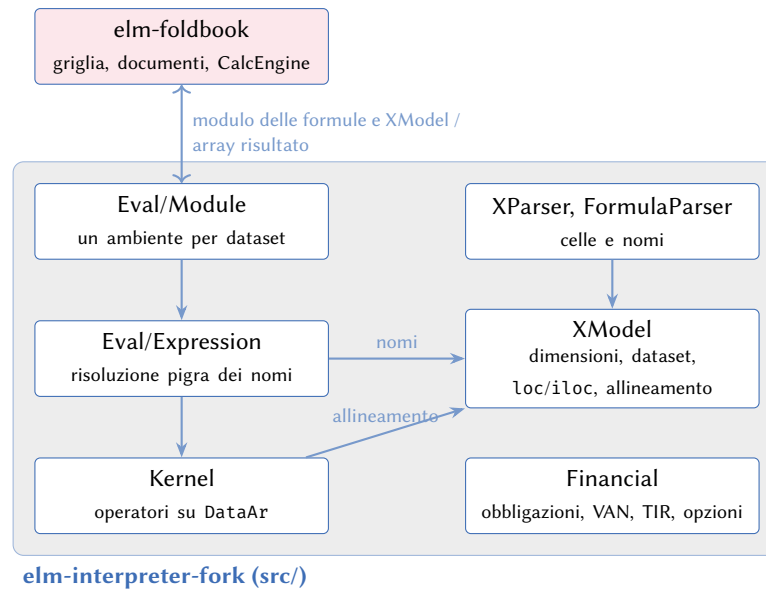


Figura 1: I moduli del fork e il loro uso da parte di elm-foldbook. Nel riquadro grigio i moduli dell'interprete estesi dall'autore (a sinistra) e i moduli nuovi (a destra); il parser di Elm e la libreria `elm/core` rigenerata, di Taglialegne, non sono rappresentati.

L'applicazione passa al motore il modello e il sorgente delle formule di un dataset. Il motore costruisce un ambiente dell'interprete per quel dataset, valuta le formule e restituisce gli array risultanti, che l'applicazione scrive nel modello (figura 1).

4.2 Il modello dati XModel

Il modello è definito in `TypesXModel.elm` e gestito in `XModel.elm` (4.145 righe). Un `XModel` contiene un dizionario di *dimensioni condivise*, ciascuna con un elenco ordinato di coordinate, e un insieme di *dataset*:

```
type alias Dataset =
  { ref : DatasetRef
  , dimRefs : List DimRef           -- dimensioni usate, in ordine
  , dataArrayRefs : List DataArrayRef -- variabili del dataset
  , dataArrays : DataArrays
  , formulas : ModuleSource         -- un modulo Elm
  , defaultDataArrayRef : Maybe DataArrayRef
  }

type alias DataArray =
  { ref : DataArrayRef
  , datasetRef : Maybe DatasetRef
  , data : Array Float              -- valori numerici
  , text : Array String             -- testi, in parallelo
  , localDims : Maybe Dims          -- dimensioni proprie di una sezione
  }
```

```
, localDimRefs : Maybe (List DimRef)
, pointedFormulas : Dict FlatIndex String
}
```

Ogni variabile (DataArray) è un array piatto in ordine *row-major*. Se un dataset ha dimensioni di lunghezza $n_1, \dots, n_k$, la cella di posizione $(i_1, \dots, i_k)$ sta all'indice

$$f = \sum_{j=1}^k i_j s_j, \quad s_j = \prod_{h>j} n_h,$$

dove s_j è il passo della dimensione j (calculateStrides, calcFlatIndexFast, flatIndexToPosVec). Le variabili di un dataset formano una pseudo-dimensione aggiuntiva, dato, che si comporta come le altre.

La selezione segue pandas e xarray: illoc seleziona per posizione, loc per coordinata, con specificatori singoli, a intervallo o a lista (CoordSingle, CoordRange, CoordList). Una sezione restituisce un nuovo DataArray con le proprie dimensioni locali. Le funzioni di modifica della struttura (appendCoord, insertCoordAt, renameDimRef, newDataArray e altre) aggiungono, inseriscono, rinominano ed eliminano coordinate, dimensioni e variabili, e riallineano i dati esistenti alla nuova forma. Il modulo contiene inoltre aggregazioni (arSum, arAverage, arStdDev, correlazioni) e il modello di esempio finPlan (XModelSample.elm).

4.3 Nomi di intervallo

Perché una formula possa nominare una porzione del modello, ogni porzione deve avere un nome che sia un identificatore Elm valido. La grammatica è

dataset__variabile_coordinata_coordinata...

per esempio ce__valore_ebitda (variabile valore del dataset Ce, coordinata ebitda della dimensione voce, tutte le aziende e tutti gli anni) o macro__cambioUsdEur_base (scenario base, tutti gli anni). Le coordinate non dicono a quale dimensione appartengono: rangeNameToDef le attribuisce con un dizionario inverso costruito da makeCoordSearchDict; se una coordinata compare in più dimensioni si qualifica come dimDOTcoord. La funzione inversa rangeDefToName ricostruisce il nome da una selezione.

promptCoordsForPartialRangeName riceve un nome scritto a metà e restituisce le coordinate che possono completarlo, escluse quelle delle dimensioni già usate: l'editor delle formule di elm-foldbook la usa per il completamento automatico (Figura 2).

Formula Editor

```

1 module Ce exposing (..)
2 -- getExprDataArray errors on taxRate, browser hangs; cyclic update may stall?!
3 -- error in getExprDataArray are overcome by parsing the module
4 -- check cycle in calcExpressionToXModel
5 mySub : Float -> Float -> Float -- parsing does not catch declaration errors
6 mySub val1 val2 = val1 - val2 -- errors are raised in range parsing
7 taxRate : Float -- declaring a constant expr is not needed, no longer parsing errors
8 taxRate = 0.4
9 -- mapping between datasets thanks to order swap in evalNonVariant before
10 -- executing Kernel.twoNumbers when a binary func is applied
11 costoVen = ce_valore_ricavi * 0.48 * macro_cambioUsdEur_base -- uses expansion of rangeName from getExprDataArray
12 ce_valore_beta_a1 _valore_ricavi ce_valore_costoVen
13 ce_valore_beta_a2 rgContrib - ce_valore_speseVGA
14 ce_valore_beta_a3 da - ce_valore_amm
15 ce_valore_beta_amm * taxRate
16 ce_valore_beta_costoVen alore_ebit - ce_valore_tax
17 ce_valore_beta_ebit
18 ce_valore_beta_ebitda
19 ce_valore_beta_maraContrib

```

Figura 2: L'editor delle formule di elm-foldbook mentre l'utente scrive `ce_valore_beta_`: le coordinate proposte (anni e voci del conto economico) sono calcolate dal fork con `promptCoordsForPartialRangeName`.

4.4 L'innesto nell'interprete

Un nuovo valore. Il tipo `Value` dell'interprete è stato esteso con il costruttore `DataAr`, che rappresenta un `dataArray` come record di valori `Elm`, insieme all'indicazione se l'array proviene da un altro dataset. L'ambiente di valutazione porta con sé il modello (`envXModel`) e le funzioni definite nel modulo delle formule.

Risoluzione pigra dei nomi. Quando il valutatore (`Eval/Expression`) incontra un nome che non è né una funzione né un valore definito, prima di segnalare l'errore prova a leggerlo come nome di intervallo. Se `rangeNameToDef` lo riconosce, il valutatore carica dal modello la variabile intera o la sua sezione (`loc`) e la restituisce come `DataAr` (estratto semplificato):

```

case XModel.rangeNameToDef xModel curDatasetRef name of
  Ok rangeDef ->
    case ( rangeDef.datasetRef, rangeDef.dataArrayRef, rangeDef.dimCoords ) of
      ( Just datasetRef, Just dataArrayRef, Just [] ) -> -- variabile intera
        XModel.getParsedDataArrayToValue ...
      ( Just datasetRef, Just dataArrayRef, Just dimCoords ) -> -- sezione
        XModel.locDataArrayfromRangeDefToValue ...

```

Così i dati entrano nell'ambiente solo quando una formula li usa, e la stessa formula può riferirsi ad altri dataset del modello.

Un ambiente per dataset. `Eval/Module` costruisce un ambiente dell'interprete per ogni dataset a partire dal suo modulo delle formule (`makeEnv`) e valuta una singola dichiarazione in quell'ambiente (`evalModuleWithEnv`), senza rileggere il sorgente a ogni ricalcolo.

4.5 Aritmetica vettoriale e allineamento

Le primitive del `Kernel` che implementano gli operatori aritmetici e le funzioni matematiche di `Basics` sono state estese a tutte le combinazioni di scalari (`Int`, `Float`) e array (`DataAr`). Con uno scalare l'operazione si applica a ogni elemento; con due array si passa alla funzione di allineamento (estratto semplificato):

```
[ DataAr ar, Float fl ] →
  let dAr = XModel.valueToArray (DataAr ar)
  in ... { dAr | data = Array.map (\x → fFloat x fl) dAr.data }

[ DataAr la, DataAr ra ] →
  ... XModel.funcDataArrayPairAr adjLeftDAR adjRightDAR fFloat
```

funcDataArrayPair confronta le dimensioni dei due operandi. Se coincidono, i dati si combinano elemento per elemento. Se differiscono, gli indici dell'operando sorgente vengono rimappati su quelli dell'operando di destinazione per coordinata (mapFlatIndices); le combinazioni che mancano nella sorgente diventano valori non numerici (NaN). Quando uno dei due array viene da un altro dataset, il risultato prende la forma dell'array del dataset corrente. È questo che permette di scrivere, nel conto economico con dimensioni azienda × voce × anno,

```
costoVen = ce__valore_ricavi * 0.48 * macro__cambioUsdEur_base
```

dove macro__cambioUsdEur_base ha la sola dimensione anno (scenario base del dataset Macro) e viene esteso a tutte le aziende.

4.6 Valori delle celle e funzioni finanziarie

XParser interpreta i valori scritti nelle celle (numero, testo, cella vuota, errore) con i separatori decimali locali; FormulaParser è una grammatica sperimentale dei nomi gerarchici. Financial contiene prezzo e rendimento a scadenza di titoli a cedola fissa e senza cedola (il rendimento con il metodo di Newton-Raphson), valore attuale netto, tasso interno di rendimento, prezzo di opzioni con Black-Scholes e Garman-Kohlhagen, e la funzione di ripartizione normale.

5 Esempi d'uso

Il fork non ha un'interfaccia propria: lo si vede all'opera in elm-foldbook. Gli esempi usano il modello finPlan di XModelSample.elm, con tre dataset: Az (anagrafica delle aziende), Ce (conto economico di due aziende in tre anni) e Macro (inflazione e cambio per scenario e anno).

5.1 Un modulo di formule

Le formule del conto economico sono un modulo Elm. Ogni dichiarazione di primo livello ha come nome l'intervallo di destinazione:

```
module Ce exposing (..)
mySub : Float → Float → Float
mySub val1 val2 = val1 - val2
taxRate : Float
taxRate = 0.4
costoVen = ce__valore_ricavi * 0.48 * macro__cambioUsdEur_base
ce__valore_margContrib = mySub ce__valore_ricavi ce__valore_costoVen
ce__valore_ebitda = ce__valore_margContrib - ce__valore_speseVGA
ce__valore_ebit = ce__valore_ebitda - ce__valore_amm
ce__valore_tax = ce__valore_ebit * taxRate
ce__valore_unlevNetIncome = ce__valore_ebit - ce__valore_tax
```

Ogni riga calcola una voce per tutte le aziende e tutti gli anni. `taxRate` è una costante e `mySub` una normale funzione Elm definita nello stesso modulo: il linguaggio di formula è l'intero linguaggio Elm. Nella valutazione di `ce__valore_ebitda` il motore risolve i due nomi a destra, carica le due sezioni `azienda` $\times$ `anno` del dataset, le sottrae elemento per elemento e restituisce un `DataAr` della stessa forma, che `elm-foldbook` scrive nella riga `ebitda`.

5.2 Ricalcolo a catena

La Figura 3 mostra l'effetto di una modifica: i ricavi dell'azienda *alfa* nel primo anno passano da 100 a 150 e il motore ricalcola tutte le voci derivate. Il costo del venduto passa da 52,8 a 79,2, l'EBITDA da 41,2 a 64,8, il reddito netto da 17,5 a 31,7. L'ordine di ricalcolo e la scrittura dei risultati nella griglia sono compiti dell'applicazione; i valori sono calcolati dal fork.

Conto economico

azienda	voce	valore			dato
		a1	a2	a3	anno
alfa	ricavi	100.0	209.0	300.0	
	costoVen	52.8	100.3	216.0	
	margContrib	47.2	108.7	84.0	
	speseVGA	6.0	17.0	18.0	
	ebitda	41.2	91.7	66.0	
	amm	12.0	29.0	16.0	
	ebit	29.2	62.7	50.0	
	tax	11.7	25.1	20.0	
	unlevNetIncome	17.5	37.6	30.0	

(a) Modello originale

Conto economico

azienda	voce	valore			dato
		a1	a2	a3	anno
alfa	ricavi	150.0	209.0	300.0	
	costoVen	79.2	100.3	216.0	
	margContrib	70.8	108.7	84.0	
	speseVGA	6.0	17.0	18.0	
	ebitda	64.8	91.7	66.0	
	amm	12.0	29.0	16.0	
	ebit	52.8	62.7	50.0	
	tax	21.1	25.1	20.0	
	unlevNetIncome	31.7	37.6	30.0	

(b) Ricavi di alfa nel primo anno portati a 150

Figura 3: Analisi *what-if* in `elm-foldbook`: le celle derivate sono ricalcolate dal motore con le formule a livello di array.

5.3 Formule fra dataset

Il dataset `Macro` contiene, per esempio,

```
module Macro exposing (..)
macro_inflazione_forecast = macro_inflazione_base * 1.02
macro_cambioUsdEur_forecast = macro_cambioUsdEur_base * 3.07
dependentDatasetRefs = [ "Ce" ]
```

La prima formula calcola l'inflazione dello scenario `forecast` per tutti gli anni a partire dallo scenario `base`. La formula del costo del venduto nel dataset `Ce`, vista sopra, usa il cambio di `Macro` e lo allinea sulla dimensione `anno`; `dependentDatasetRefs` dichiara che `Ce` va ricalcolato dopo `Macro`.

6 Limiti e sviluppi

6.1 Limiti

- **Ordine di calcolo.** Il motore valuta una formula alla volta; l'ordine è quello di dichiarazione e le dipendenze fra dataset sono dichiarate a mano. Non c'è un grafo delle dipendenze e i riferimenti circolari non sono gestiti.
- **Tipo dei valori.** `DataAr` rappresenta l'array come record di valori Elm, convertito da e verso `DataArray` a ogni operazione: una soluzione semplice da innestare nell'interprete, ma costosa e non controllata dal sistema dei tipi.
- **Prestazioni.** Interpretare Elm nel browser è adatto a modelli piccoli.

- **Copertura.** Le aggregazioni di XModel non sono ancora esposte alle formule; i valori mancanti nell'allineamento diventano NaN senza segnalazione; i messaggi d'errore sono rudimentali.
- **Pacchetto.** Il fork non è pubblicato come pacchetto Elm: `elm-foldbook` ne compila i sorgenti.

6.2 Che cosa ne è seguito

Il lavoro sul fork fra dicembre 2024 e marzo 2025 non è stato pubblicato e non fa parte del prodotto. Le idee e i tipi del modello sono passati a Impromptu (Erzegovesi 2026b), l'ambiente di modellazione multidimensionale che l'autore sviluppa dal febbraio 2026: il front end Elm di Impromptu contiene fin dal primo commit (22/02/2026) i tipi `XModel`, `Dataset`, `dataArray` e le funzioni di selezione derivati dal fork, mentre il calcolo è affidato a un server Julia con grafo delle dipendenze e un linguaggio di formula a livello di array compilato.

7 Disponibilità, licenza e riproducibilità

7.1 Dove si trova

- Repository: <https://github.com/lerzegov/elm-interpreter-fork>, pubblico dal 29/07/2024; versione di riferimento il commit `bc148ee` del 18/12/2024, ultimo push.
- Zenodo (Erzegovesi 2024): DOI 10.5281/zenodo.23056240, archivio `git archive` del commit `bc148ee` depositato il 30/09/2026 senza modifiche, con data di pubblicazione 18/12/2024. Il contenuto coincide con la directory Software Heritage `swh:1:dir:8498297b9e38243773513bac9ba694857f789c5a`.
- Software Heritage: `swh:1:rev:bc148ee3cdb33f435fd68f575a508c01444768ea`. L'identificativo deriva dal contenuto e coincide con l'hash `git` del commit.
- Il record Zenodo di `elm-foldbook` (Erzegovesi 2026a) dichiara di richiedere questo prodotto (*Requires*).

7.2 Licenza

BSD 3-Clause, la stessa di `elm-interpreter`. Leonardo Taglialegne, autore dell'interprete, è stato informato della pubblicazione e non ha sollevato obiezioni (30/09/2026).

7.3 Compilare ed eseguire

Il fork si usa compilandolo insieme a un'applicazione Elm. `make` rigenera con `elm-codegen` la libreria `elm/core` per l'interprete (cartella `generated/`); i test dell'interprete si eseguono con `elm-test`. Il modo più diretto per vederlo funzionare è `elm-foldbook`: il suo rilascio 0.7.1 include in `vendor/elm-interpreter-fork` lo stato del motore con cui la v0.7 è stata compilata il 16/12/2024, ricostruito e verificato sul programma compilato conservato nel repository (la procedura è descritta nella nota di `elm-foldbook` e in `vendor/README.md`).

Ringraziamenti

Il fork si fonda su `elm-interpreter` di Leonardo Taglialegne, che l'autore ringrazia anche per il confronto durante lo sviluppo.

Uso di assistenti di intelligenza artificiale. Alcune parti del codice del 2024 sono state scritte con l'aiuto di ChatGPT, come segnalato nei commenti del sorgente. La stesura di questa nota è stata assistita da Claude (Anthropic). Scelte progettuali, contenuti e verifiche sono dell'autore.

Riferimenti bibliografici

- Czaplicki, Evan, e Stephen Chong. 2013. «Asynchronous Functional Reactive Programming for GUIs». *Proceedings of the 34th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, 411–22.
- Erzegovesi, Luca. 2024. *elm-interpreter-fork: multidimensional data model (XModel) and array evaluation for elm-interpreter, calculation engine of elm-foldbook*. Zenodo, released. <https://doi.org/10.5281/zenodo.23056240>.
- Erzegovesi, Luca. 2026a. *elm-foldbook: a multidimensional spreadsheet prototype with mathematical documents for teaching and research in corporate finance*. Zenodo, released. <https://doi.org/10.5281/zenodo.23056100>.
- Erzegovesi, Luca. 2026b. «Impromptu under the hood». The Workbench. <https://theworkbench.lerzegov.org/posts/impromptu/>.
- Hoyer, Stephan, e Joe Hamman. 2017. «xarray: N-D Labeled Arrays and Datasets in Python». *Journal of Open Research Software* 5 (1): 10.
- Iverson, Kenneth E. 1962. *A Programming Language*. Wiley.
- McKinney, Wes. 2010. «Data Structures for Statistical Computing in Python». *Proceedings of the 9th Python in Science Conference*, 56–61.
- Peyton Jones, Simon, Alan Blackwell, e Margaret Burnett. 2003. «A User-Centred Approach to Functions in Excel». *Proceedings of the Eighth ACM SIGPLAN International Conference on Functional Programming (ICFP)*, 165–76.
- Quantrix. s.d. *Quantrix Modeler*. <https://www.quantrix.com>.
- Stil4m. s.d. *elm-syntax*. <https://package.elm-lang.org/packages/stil4m/elm-syntax/latest/>.
- Taglialegne, Leonardo. 2023. *elm-interpreter*. Released. <https://github.com/miniBill/elm-interpreter>.

A Moduli del fork

LE = L. Erzegovesi, LT = L. Taglialegne. Righe alla revisione bc148ee.

Tabella 3: Moduli del fork.

Modulo	Righe	di LE	Origine	Funzione
XModel	4.145	4.145	LE	Modello multidimensionale, selezione, struttura, allineamento, nomi di intervallo
TypesXModel	957	957	LE	Tipi del modello, vassoi, intestazioni, coordinate
XModelSample	624	624	LE	Modello di esempio finPlan
XParser	105	105	LE	Valori delle celle

Modulo	Righe	di LE	Origine	Funzione
FormulaParser	89	89	LE	Nomi gerarchici (sperimentale)
Financial	116	116	LE	Funzioni finanziarie
AppUtil, XModelHelpers	290	290	LE	Utilità
Kernel	869	163	LT + LE	Primitive; LE: operatori e funzioni su array
Eval/Expression	1.406	201	LT + LE	Valutatore; LE: nomi di intervallo, operazioni su array
Eval/Module	321	211	LT + LE	Moduli; LE: ambienti per dataset con XModel
Types, Value, Environment, Eval, Eval/Types, Kernel/Utils, UI, UI/Theme	1.878	201	LT + LE	Tipo DataAr, ambiente con il modello
UI/Source, TopologicalSort, EvalResult, Syntax e altri	1.090	–	LT	Interprete originale
codegen/, helpers/, generated/	–	–	LT	elm/core rigenerata per l'interprete