CALORIE COUNTING WITH JACQUARD

Brian Beckman
6 June 2012

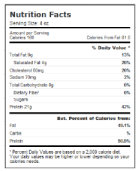
- **Essential contributions from**
  Avi Bar-Zeev, Elad Ben-Israel, Steve Coast, Elad Gerson, Yair Gheva, Gur Kimchi, Limor Lahiani, Erik Meijer, Kanchan Mitra, Savas Parastiditis

- **Sponsored by IPE Core Platform**
  Bimal Mehta, James Whittaker, Gurdeep Singh Pall, Qi Lu

MOTIVATING SAMPLE PROBLEM

- **How to write apps to manipulate standard Nutrition-Facts Labels (NFLs)?**
- **e.g.: (1) Check Consistency (2) Compute new labels from recipes on-the-fly**
- **Jacquard enables computations that are *impossible via ordinary programming* in JavaScript or C# and *easy with symbolic expressions***

• Problems with NFLs:



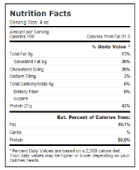
• Confusing information; confusing layout

• Many units: ounces, percents, calories, grams, milligrams

• Separate calories from fat & carbs, not percents

• Check for accuracy: Does serving size and total calories match sum of components? If not, why not? Liability exposure?

• Problems with NFLs:

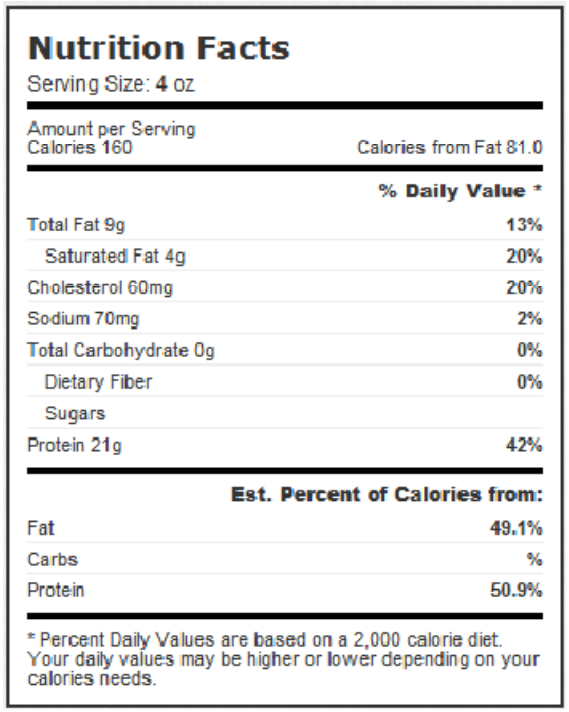


• Confusing information; confusing layout

• Many units: ounces, percents, calories, grams, milligrams

• Separate calories from fat & carbs, not percents

• Check for accuracy: Does serving size and total calories match sum of components? If not, why not? Liability exposure?



Nutrition Facts

Serving Size: **4** oz

Amount per Serving
Calories 160 — Calories from Fat 81.0

	% Daily Value *
Total Fat 9g	13%
Saturated Fat 4g	20%
Cholesterol 60mg	20%
Sodium 70mg	2%
Total Carbohydrate 0g	0%
Dietary Fiber	0%
Sugars	
Protein 21g	42%

Est. Percent of Calories from:	
Fat	49.1%
Carbs	%
Protein	50.9%

* Percent Daily Values are based on a 2,000 calorie diet. Your daily values may be higher or lower depending on your calories needs.

- Problems with NFLs:
- Confusing information; confusing layout
- Many units: ounces, percents, calories, grams, milligrams
- Separate calories from fat & carbs, not percents
- Check for accuracy: Does serving size and total calories match sum of components? If not, why not? Liability exposure?

WHAT IS JACQUARD? http://bing/wiki/Jacquard

- **General-purpose, platform-indep, distributed, sandboxed Expression Evaluator**
- **Value prop: safe, small, mobile, distributed, compatible, reusable scripting**
- **Expression Evaluators run everywhere: clients, servers, services**
- **multistyle: Symbolic, Numeric, Logic, Functional, Object**
- **JSON, AJAX, Node.Js (try http://jacquard.msproto.net/)**
- **(soon) .NET LINQ Expressions and Native code**
- **Established techniques -- low "science" risk**
- **Pattern Matching -- "Regular Expressions" for whole expressions, not just strings**
- **Term Rewriting: "*when you match this pattern, replace it with that expression after substituting values for variables*"**
  -- ordinary function calls are hard-coded rewrite rule
  -- TR is more general: you write your own rewrite rules
- **Built-in to the Knowledge Store from Day Zero: Expressions are Entities,**
  Fully self-describing and reflective

USES OF GENERAL COMPUTATION

- **Most searches already involve computations**
  - Route plans, Travel proposals
  - Event-sequence schedules
  - Price quotes, loan proposals
  - Trading, portfolio optimization, risk analysis
  - Even calorie counting
- **Jacquard makes computations easier to write, more robust, distributed, reusable**
  - Keep Computations and other Knowledge **in exactly the same format**
  - Computations are not arbitrary programs; sanboxing is easier
  - Index and retrieve **expressions as Merino Entities**
  - **More computation for less script**
- **Wolfram Alpha blazed the trail**
  - Mission statement similar to Merino's
  - They are 100% based on an expression language -- *Mathematica* -- which inspired Jacquard
  - They curate; we crowdsource

THE SCENARIO: WHERE'S THE BEEF?

- **Encode Nutrition-Fact Label in JavaScript (C# similar); contrast Jacquard:**

JavaScript

```
var burgerNutritionFacts =
{ ServingSize       :   4   , // Ounce
  AmountPerServing  : 160   , // Calorie
  CaloriesFromFat   :  81.0, // Calorie
  SaturatedFat      :   4   , // Gram
  Cholesterol       :  60   , // Milligram
  Sodium            :  70   , // Milligram
  DietaryFiber      :   0   , // Gram
  Sugars            :   0   , // Gram
  TotalFat          :   9   , // Gram
  Protein           :  21   , // Gram
  TotalCarbohydrate :   0   , // Gram
};
```

Jacquard

```
burgerNutritionFacts = {
  ServingSize       ->   4   * Ounce,
  AmountPerServing  -> 160   * Calorie,
  CaloriesFromFat   ->  81.0 * Calorie,
  SaturatedFat      ->   4   * Gram,
  Cholesterol       ->  60   * Milli Gram,
  Sodium            ->  70   * Milli Gram,
  DietaryFiber      ->   0   * Gram,
  Sugars            ->   0   * Gram,
  TotalFat          ->   9   * Gram,
  Protein           ->  21   * Gram,
  TotalCarbohydrate ->   0   * Gram
}
```

- **OBSERVATION: In JavaScript, NO innate way to carry units**
- **Imagine this is written by OCR program, not by hand**
- **Jacquard: *4 * Ounce* means "4 times the symbolic constant *Ounce*"**
- **Jacquard: *Everything is an Expression* -- No Exceptions**
- **Computing in Jacquard is Evaluating Expressions, not "running code"**

DO THE WEIGHTS ADD UP?

JavaScript	Jacquard

`var addWeights = function(nutritionFacts) {`
`return nutritionFacts.TotalFat +`
`nutritionFacts.DietaryFiber +`
`nutritionFacts.Protein +`
`nutritionFacts.Cholesterol +`
`nutritionFacts.Sodium +`
`nutritionFacts.TotalCarbohydrate;`
`};`
`document.writeln(addWeights(burgerNutritionFacts));`
	`TotalFat +`
	`DietaryFiber +`
	`Protein +`
	`Cholesterol +`
	`Sodium +`
	`TotalCarbohydrates`
	`/. burgerNutritionFacts`

`160`	`30 Gram + 130 Gram Milli`

- **NOTE: Jacquard catches the mistake!**
  In JavaScript, only the programmer can catch it, and only by head compiling
- **NOTE: this 160 is highly suspect: will see later**
- **NOTE: Units mistakes can cost billions**
  - 1999: NASA crashed Mars Climate Observer over force in Newtons versus pounds-force
  - Who would be liable for a faulty NFL implicated in a diabetic or cardiac incident?
- **OBSERVATION: object access (slash-dot) *after* the arithmetic, not before**
- **Object access distributes across the sum**

HOW DOES IT WORK?

- **Not like ordinary programming; *unbound variables are NOT errors***
- **Symbolic constants are like variables that just evaluate to themselves**
- **They cancel out of ratios ...**

$$\frac{16\ \text{Ounce}}{\text{Pound}} * 27\ \text{Pound}$$

```
432 Ounce
```

- **... and distribute over sums**

```
2 Ounce + 4 Ounce
```

```
6 Ounce
```

APPLYING OBJECTS TO EXPRESSIONS

- **This is a symbolic expression; it evaluates to itself:**

```
TotalFat + DietaryFiber + Protein +
  Cholesterol + Sodium + TotalCarbohydrate
```

```
Cholesterol + DietaryFiber + Protein + Sodium + TotalCarbohydrate + TotalFat
```

- **Save it in a variable ...**

```
nflSummary = TotalFat + DietaryFiber + Protein +
  Cholesterol + Sodium + TotalCarbohydrate
```

```
Cholesterol + DietaryFiber + Protein + Sodium + TotalCarbohydrate + TotalFat
```

JACQUARD OBJECT = LIST OF REPLACEMENT RULES

```
(burgerNutritionFacts = {ServingSize → 4 * Ounce,
    AmountPerServing → 160 * Calorie,
    CaloriesFromFat → 81.0 * Calorie,
    SaturatedFat → 4 * Gram, Cholesterol → 60 * Milli * Gram,
    Sodium → 70 Milli * Gram, DietaryFiber → 0 * Gram,
    Sugars → 0 * Gram, TotalFat → 9 * Gram, Protein → 21 * Gram,
    TotalCarbohydrate → 0 * Gram}) // gridRules
```

ServingSize	Times 4 Ounce
AmountPerServing	Times 160 Calorie
CaloriesFromFat	Times 81. Calorie
SaturatedFat	Times 4 Gram
Cholesterol	Times 60 Gram Milli
Sodium	Times 70 Gram Milli
DietaryFiber	0
Sugars	0
TotalFat	Times 9 Gram
Protein	Times 21 Gram
TotalCarbohydrate	0

APPLY THE OBJECT TO THE EXPRESSION

```
nflSummary   /.   burgerNutritionFacts
```

```
30 Gram + 130 Gram Milli
```

- **Long form of the same expression**

```
ReplaceAll[ nflSummary, burgerNutritionFacts ]
```

```
30 Gram + 130 Gram Milli
```

- **Objects are collections of replacement rules**
- ***Applying rules* is a generalization of *calling functions***
- **Objects act like collections of functions**
  - This is also true in ordinary object-oriented programming
  - Methods are always functions
  - Properties are backed by functions (get, set) in ECMAScript & C#

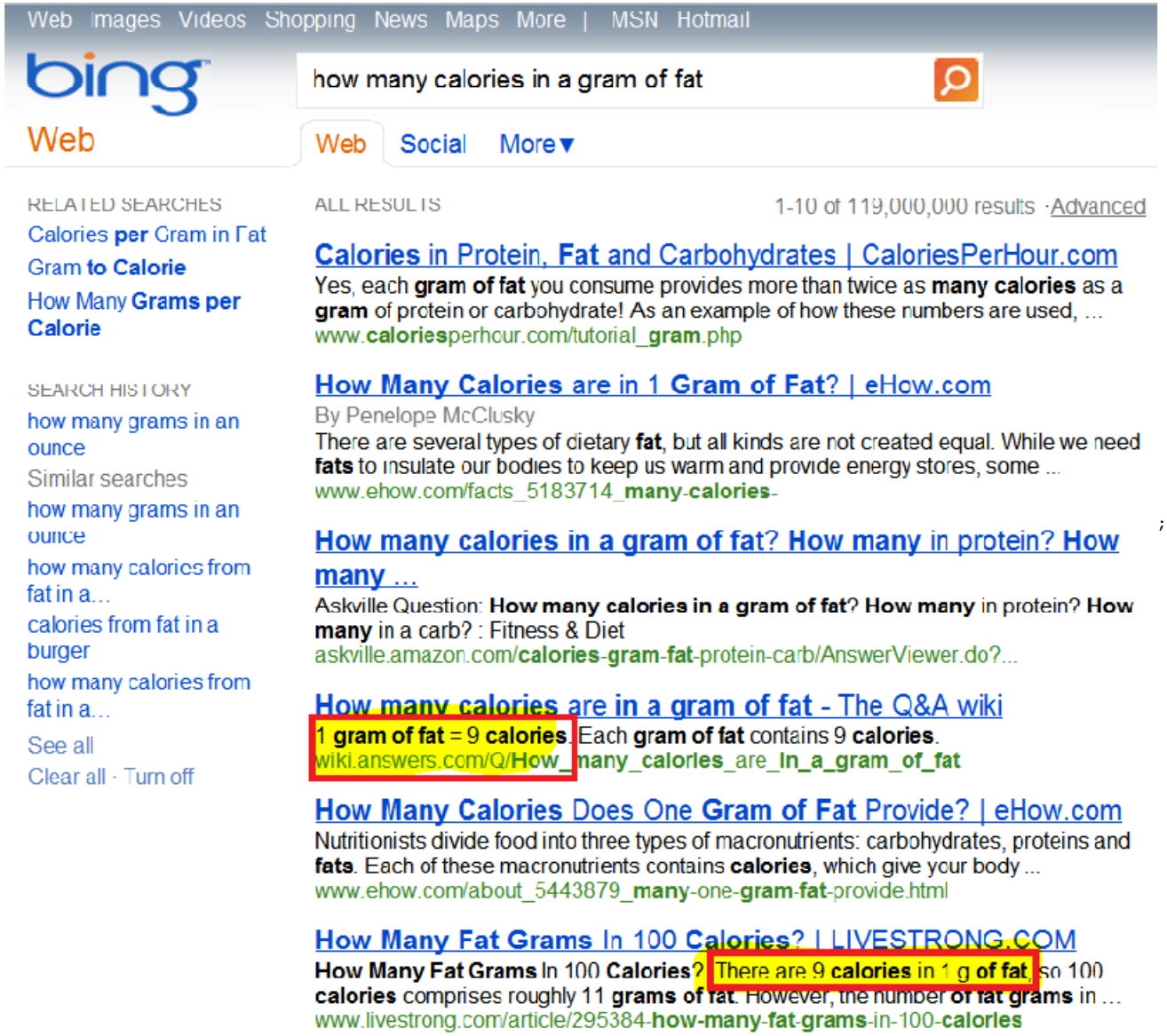
MORE NFL: ADD UP THE CALORIES

- **Beef-up the object**

```
(beefedUpBurgerNutritionFacts = {ServingSize → 4 * Ounce,
    AmountPerServing → 160 * Calorie, CaloriesFromFat → 81.0 * Calorie,
    SaturatedFat → 4 * Gram * saturated fat,
    Cholesterol → 60 * Milli * Gram * cholesterol,
    Sodium → 70 Milli * Gram * sodium, DietaryFiber → 0 * Gram * fiber,
    Sugars → 0 * Gram * sugar, TotalFat → 9 * Gram * fat,
    Protein → 21 * Gram * protein,
    TotalCarbohydrate → 0 * Gram * carbohydrate}) // gridRules
```

ServingSize	Times 4 Ounce
AmountPerServing	Times 160 Calorie
CaloriesFromFat	Times 81. Calorie
SaturatedFat	Times 4 fat Gram saturated
Cholesterol	Times 60 cholesterol Gram Milli
Sodium	Times 70 Gram Milli sodium
DietaryFiber	0
Sugars	0
TotalFat	Times 9 fat Gram
Protein	Times 21 Gram protein
TotalCarbohydrate	0

MINE FOR CALORIE FACTS



ENCODE CALORIE FACTS AS RULES

```
(calorieFacts = {
    Gram * saturated * fat → 9 * Calorie,
    Gram * fat → 9 * Calorie,
    Gram * sugar → 4 * Calorie,
    Gram * carbohydrate → 4 * Calorie,
    Gram * protein → 4 * Calorie,
    Gram * cholesterol → 0 * Calorie,
    Gram * fiber → 0 * Calorie,
    Gram * sodium → 0 * Calorie,
    Milli * Gram → Gram * 0.001}) // gridRules
```

fat Gram saturated	Times 9 Calorie
fat Gram	Times 9 Calorie
Gram sugar	Times 4 Calorie
carbohydrate Gram	Times 4 Calorie
Gram protein	Times 4 Calorie
cholesterol Gram	0
fiber Gram	0
Gram sodium	0
Gram Milli	Times 0.001 Gram

APPLY CALORIE FACTS & SUMMARIZE

```
(beefedUpBurgerNutritionFacts /. calorieFacts) // gridRules
```

ServingSize	Times 4 Ounce
AmountPerServing	Times 160 Calorie
CaloriesFromFat	Times 81. Calorie
SaturatedFat	Times 36 Calorie
Cholesterol	0
Sodium	0
DietaryFiber	0
Sugars	0
TotalFat	Times 81 Calorie
Protein	Times 84 Calorie
TotalCarbohydrate	0

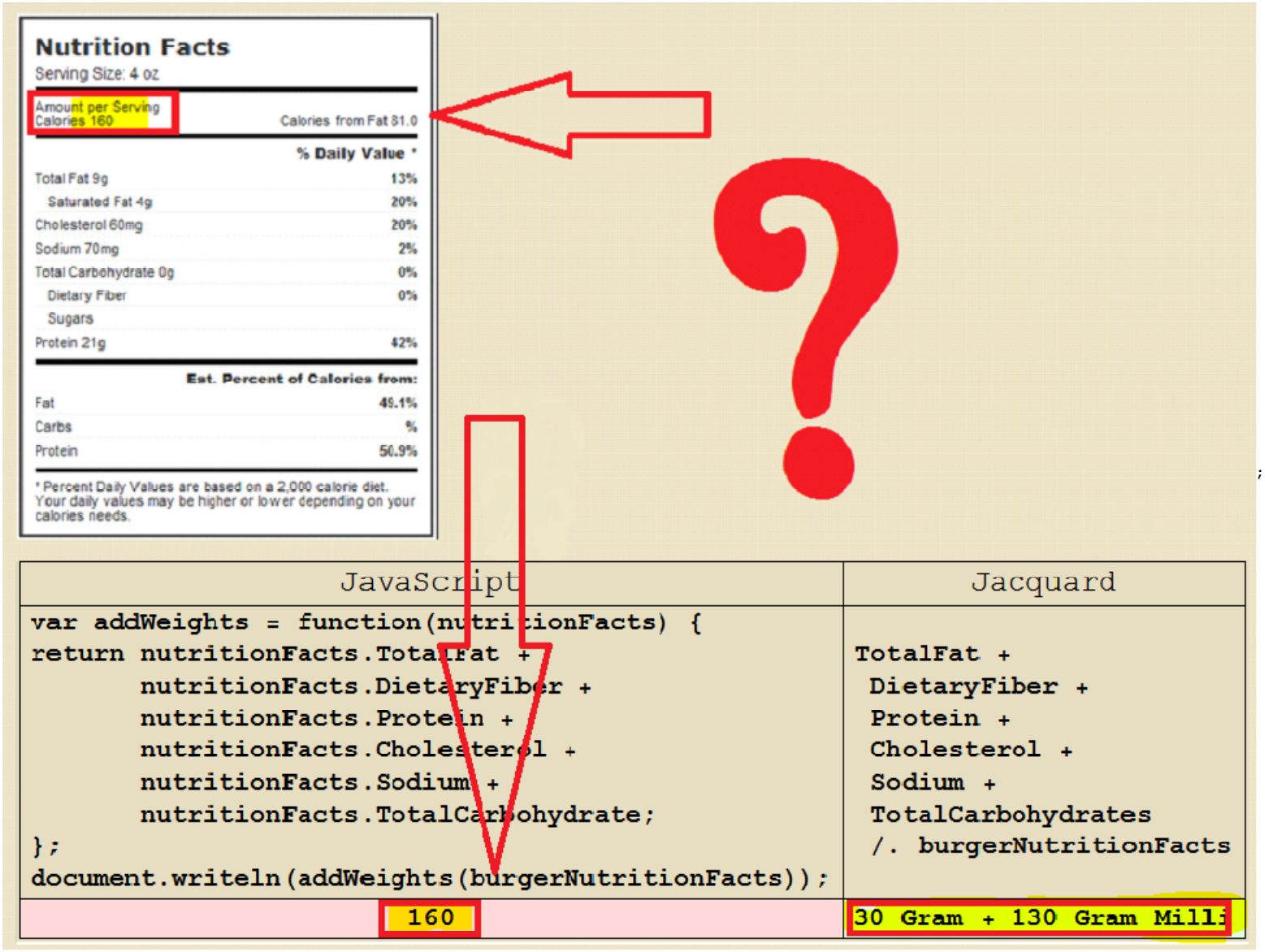
- **This is just a new object: APPLY IT to our first sum expression!**

```
nflSummary /. beefedUpBurgerNutritionFacts /. calorieFacts
```

```
165 Calorie
```

- **but but but -- they said 160 Calorie? What's up with *that*?**

UH-OH! VERY SUSPECT 160 HERE



ORIGINAL AUTHORS MADE A MISTAKE?

- **Data adds up to 165 calories; NFL reports 160**
- **We earlier found 160, without units, in a wrong weight calculation**
- **Hypothesis: Programmers made the obvious mistake and then *also* copied the bad weight output into the wrong calorie slot**
- **Without units, it's probably an honest mistake, but too easy to make and compound**
- **If we carry units, this kind of mistake requires willful malfeasance**
  - Symbolic expressions make the mistake nearly impossible to miss

- **NASA recommended research in units processing after the Mars crash**
  - Commercial manifestations include Sun's FORTRESS language and .NET's F#
  - Symbolic programs like Macsyma, Maple and *Mathematica* have had units of measure for decades

IT'S WORSE!

- **We found 30 Gram + 130 Gram Milli**
- **Check against serving size -- use original weight-finding expressions:**

```
Convert[ nflSummary /. burgerNutritionFacts, Ounce ]
```

```
1.0628 Ounce
```

- **NFL says serving size = 4 ounces: almost 3 ounces of MISSING MASS?**
- **Willful underreporting?**
- **Unreported inert ingredients like water?**
- **We can't say from the data!**
- **Can we trust the Calories per Serving results?**
- **BOTTOM LINE: Jacquard gives the IPE Developer clear opportunity to write safer, smaller, more disciplined (i.e., better), apps**

TAKEAWAYS

- **Computing with Expressions helps your apps code**
  - short, clear, expressive, sandboxable
- **Authoring environments in *Mathematica* now and (planned) Cloud9**
- **Interop with JavaScript (now), C#, C++ (planned)**
- **Reactive & Enumerative LINQ is a Given**
- **Expressions are just more knowledge in the graph**
- **Expressions are universal**
  - Supports Symbolic, Numeric, Logic, Functional, Object models
- **Evaluators bring Expressions to life**
  - Evaluators everywhere: agent, cloud, desktop, edge, all devices
  - Reference Evaluator in JavaScript; more evaluators in C#/F#, Java/Scala/Kiama, C++/Maude

EXPRESSIONS AS ENTITIES

- **Encode in JSON**

```
nflSummary//rulesFromExpression//gridRules
```

head	symbol	Plus
parts	symbol	Cholesterol
	symbol	DietaryFiber
	symbol	Protein
	symbol	Sodium
	symbol	TotalCarbohydrate
	symbol	TotalFat

- **Replace symbols with URIs in the Expressions Taxonomy / Ontology:**

```
nflSummary/.{Plus→
{"$meta"→{"knol"→"knol:knowledge.merino.com/",
  "expressions"→"knol:expressions.merino.com/"},
  "knol_identity"→"knol:expressions.merino.com/WellKnown/Plus",
  "knol_types/is-a"→"knol:expressions.merino.com/types/builtIn",
  "expressions_builtIn/name"→"Plus","expressions_builtIn/Attributes"→
    {"Flat", "Listable","NumericFunction","OneIdentity","Orderless","Protected",
    {"Default"→"knol:expressions.merino.com/values/builtIn/Integers/Constants/Zero"}},
  "expressions_builtIn/Doclet"→"knol:music.merino.com/expressions/Doclet/Plus"}};
```

ONCE IN MERINO

- **Index, Find, Abstract, Compose new Expressions**
- **Self-describing at all levels**
- **Grow Expression store by crowdsourcing computations**
- **Evaluate close to the data**
  - for **Privacy** (intelligent agent)
  - for **Perf** (IOC / Reactive framework)
  - for **Affinity** to data sources (RESTLINQ & bandwidth saving)
- **Semantic queries enabled by *Abstract Query DSLs***
  - Example: you want to build a computation that computes *the average age of singers in the US*
  - This is similar to a computation of *maximum salaries of CEOs in Kentucky*
  - Abstract Query DSLs find the most abstract form of the query
- **Create Abstract Query DSLs automatically from BNFs of specialized computations**

OTHER SCENARIOS

- **Get Me to the Airport on Time**
  - **Reactive** LINQ, distributed example monitoring traffic, current location, and flight status
- **Find me a funny movie that my friend Martha also likes**
  - Distributed **conversational workflow** with unification
  - Graph mashup
- **What is the Average Age of Pop Singers in the US?**
  - Example of **map-reduce style** queries
- **"If you Drive Out of Your Way, I'll Give You a Discount"**
  - **Partial-trust, geospatial**, secret auction process

- **We need more -- send us your ideas!**

BONUS ROUND: COMPUTING LABELS ON-THE-FLY

- **What would the NFL be for your mom's Pasta Primavera recipe?**
- ***Only you* know the recipe; no point searching**
- **Compute it on-the-fly by adding up the NFLs of the ingredients**
- **NFLs for real-world recipes could be a whole business built on Merino**

- **GEEKNOTE: This is a vector-space sum: unit vectors are the NFLs for the ingredients, coefficients are the amounts from your recipe**

YOUR SECRET RECIPE FROM MOM

```
myRecipe={
 1.0 Tablespoon "olive oil",
 16.0 Ounce "zucchini",
 3.5 Teaspoon "salt",
 1.5 Pound "eggplant",
 1.0 "onion",
 2.0 "bell pepper",
 14.5 Ounce "stewed tomato",
 0.5 Teaspoon "black pepper",
 0.5 Teaspoon "dried basil",
 0.5 Teaspoon "sugar",
 12.0 Ounce "pasta",
 0.25 Cup "parmesan cheese"};
```

▪ FINAL RECIPE, FEEDING SIX

```
scaleNfl[Fold[sumNfls, First[scaledNfls], Rest[scaledNfls]], 1 / 6] // gridRules
```

name	Plus	bell pepper	black pepper	dried basil	eggplant	olive oil	onion	parmesan cheese	pasta	salt	stewed tomato	sugar	zucchini
serving size	Times 368.02 Gram												
total calories	Times 309.72 Calorie												
fat calories	Times 58.413 Calorie												
total fat	Times 6.546 Gram												
% daily total fat	Times 9.8998 Percent												
saturated fat	Times 1.5959 Gram												
% daily saturated fat	Times 7.5358 Percent												
trans fat	0												
cholesterol	Times 44.422 Gram												
% daily cholesterol	Times 14.795 Percent												
sodium	Times 1.7696 Gram												
% daily sodium	Times 73.6 Percent												
total carbohydrates	Times 53.543 Gram												
% daily carbohydrates	Times 17.71 Percent												
dietary fiber	Times 6.8463 Gram												
%daily dietary fiber	Times 25.73 Percent												
sugars	Times 5.8525 Gram												
protein	Times 12.1 Gram												
% daily protein	Times 2.7658 Percent												
vitamin A	Times 13.434 Percent												
vitamin C	Times 107.62 Percent												
calcium	Times 11.903 Percent												
iron	Times 19.675 Percent												

JACQUARD FOR THIS APP

- **PLAN:**

1. Convert recipe volumes and weights into standardized weights in grams:

```
(recipeInGrams =
   Map[Function[ingredient, Convert[ingredient, Gram]], myRecipe /. volumeRules /. wholeItemRules]) // gridRules
```

Times	12.713	olive oil	Gram
Times	453.59	zucchini	Gram
Times	19.915	salt	Gram
Times	680.39	eggplant	Gram
Times	151.2	onion	Gram
Times	113.4	bell pepper	Gram
Times	411.07	stewed tomato	Gram
Times	1.05	black pepper	Gram
Times	0.5	dried basil	Gram
Times	2.1	sugar	Gram
Times	340.19	pasta	Gram
Times	22.	parmesan cheese	Gram

2. Data-mine basic NFLs for the ingredients

name	black pepper
serving size	1. Tablespoon
total calories	16 Calorie
fat calories	2 Calorie
total fat	0
% daily total fat	0
saturated fat	0
% daily saturated fat	0
trans fat	0
cholesterol	0
% daily cholesterol	0
sodium	3 Gram Milli
% daily sodium	0
total carbohydrates	4 Gram
% daily carbohydrates	1. Percent
dietary fiber	2 Gram
%daily dietary fiber	7 Percent
sugars	0
protein	1. Gram
% daily protein	0
vitamin A	0
vitamin C	2 Percent
calcium	3 Percent
iron	10 Percent

3. Normalize ingredient NFLs into “per-gram” from “per serving size”

name	black pepper
serving size	1. Gram
total calories	2.5397 Calorie
fat calories	0.31746 Calorie
total fat	0.
% daily total fat	0.
saturated fat	0.
% daily saturated fat	0.
trans fat	0.
cholesterol	0.
% daily cholesterol	0.
sodium	0.00047619 Gram
% daily sodium	0.
total carbohydrates	0.63492 Gram
% daily carbohydrates	0.15873 Percent
dietary fiber	0.31746 Gram
%daily dietary fiber	1.1111 Percent
sugars	0.
protein	0.15873 Gram
% daily protein	0.
vitamin A	0.
vitamin C	0.31746 Percent
calcium	0.47619 Percent
iron	1.5873 Percent

4. Multiply each normalized NFL by the recipe amount in grams
5. Add up all the NFLs, slot-by-slot

- **1. CONVERT RECIPE VOLUMES AND WEIGHTS INTO GRAMS**
- **Data-mine for density facts**

```
density["olive oil"] = Mean[{6.68,7.67}] * Pound / Gallon;
density["salt"] = 5.69 Gram / Teaspoon;
density["black pepper"] = 2.1 Gram / Teaspoon;
density["dried basil"] = 1.0 Gram / Teaspoon;
density["sugar"] = 4.2 Gram / Teaspoon;
density["parmesan cheese"] = 88 Gram / Cup;

density[___] = 1.0;
```

- **This is in the form of a standard Merino knowledge base:**
- **Predicate (density) assertions about entities (“olive oil”)**

■ COMPUTE GRAMS PER TARGET VOLUME FROM DENSITY
args: [TargetVolume (*e.g.Tablespoon*), DataMinedDensity]

- If you match a *target volume* (say, Tablespoon) and a data-mined density of the form *d(number)* times *weight(symbol)* divided by *volume(symbol)*, replace the match with the computed weight in grams

```
gramPerTargetVolumeFromDensity[
  targetVolume_,
  d_?NumberQ * weight_/volume_] :=

 (d * Convert[weight, Gram]) / Convert[volume, targetVolume]
```

■ WEIGHT RULE FROM QUANTIFIED INGREDIENT VOLUME
args: [Volume *e.g. 4 Teaspoon*]

- This is **a rule that creates new rules**!, i.e., an *object factory*
- If you match a *quantified ingredient volume* of the form *quantity(number)* times *ingredient(symbol)* times *volume(one of a fixed collection of symbols)*, then replace the match with a new RULE that will convert the weight of the quantified ingredient volume into grams
- Add a default rule that produces the empty list for any other match since we use this in LINQ's *SelectMany*

```
weightRuleFromQuantifiedIngredientVolume[quantity_?NumberQ  *  ingredient_  *volume : (Teaspoon | Tablespoon | Cup | FluidOunce | Pint | Gallon)] :=

  ingredient * volume →
   ingredient * gramPerTargetVolumeFromDensity[volume,
     density[ingredient]] * volume;

weightRuleFromQuantifiedIngredientVolume[___] := {}
```

VOLUME RULES

- The original recipe is an object containing quantified ingredient volumes
- Convert the recipe into a new object that specifies all weights in Grams
- *SelectMany* is regular critter for object-to-object transforms

```
(volumeRules = SelectMany[myRecipe, weightRuleFromQuantifiedIngredientVolume]) // gridRules
```

olive oil Tablespoon	Times 12.713 olive oil Gram
salt Teaspoon	Times 5.69 salt Gram
black pepper Teaspoon	Times 2.1 black pepper Gram
dried basil Teaspoon	Times 1. dried basil Gram
sugar Teaspoon	Times 4.2 sugar Gram
parmesan cheese Cup	Times 88 parmesan cheese Gram

WEIGHT RULE FOR WHOLE-ITEM INGREDIENTS

```
wholeItemWeight["onion"] = (1.0 / 3) Pound;
wholeItemWeight["bell pepper"] = 0.5 Pound / 4;


weightRuleFromQuantifiedWholeItemIngredient[
   Except[_ * _String * _Symbol, (* don't match a triple rule *)
    _?NumberQ * ingredient_ (* do match a pair *)]] :=
  (* generate the following rule *)
  ingredient → ingredient * wholeItemWeight[ingredient];
weightRuleFromQuantifiedWholeItemIngredient[___] = {};
```

WHOLE-ITEM RULES

```
(wholeItemRules = SelectMany[myRecipe, weightRuleFromQuantifiedWholeItemIngredient]) // gridRules
```

onion	Times	0.33333	onion	Pound
bell pepper	Times	0.125	bell pepper	Pound

■ RECIPE WITH EVERYTHING AS A WEIGHT

```
(myRecipe /. volumeRules /. wholeItemRules) // gridRules
```

Times	12.713	olive oil	Gram
Times	16.	zucchini	Ounce
Times	19.915	salt	Gram
Times	1.5	eggplant	Pound
Times	0.33333	onion	Pound
Times	0.25	bell pepper	Pound
Times	14.5	stewed tomato	Ounce
Times	1.05	black pepper	Gram
Times	0.5	dried basil	Gram
Times	2.1	sugar	Gram
Times	12.	pasta	Ounce
Times	22.	parmesan cheese	Gram

RECIPE IN GRAMS

```
(recipeInGrams =
   Map[Function[ingredient, Convert[ingredient, Gram]], myRecipe /. volumeRules /. wholeItemRules]) // gridRules
```

Times	12.713	olive oil	Gram
Times	453.59	zucchini	Gram
Times	19.915	salt	Gram
Times	680.39	eggplant	Gram
Times	151.2	onion	Gram
Times	113.4	bell pepper	Gram
Times	411.07	stewed tomato	Gram
Times	1.05	black pepper	Gram
Times	0.5	dried basil	Gram
Times	2.1	sugar	Gram
Times	340.19	pasta	Gram
Times	22.	parmesan cheese	Gram

Just for Fun

```
Convert[Apply[Plus, Cases[recipeInGrams, q_ * _String * u_Symbol → q u]], Pound]
```

```
4.8681 Pound
```

atsa lotta pasta -- serves at least six

2. & 3. UNIT-NFLS FOR ALL INGREDIENTS

```
ClearAll[nfls,nflNames]; nflNames={};
```

- Just a constructor that builds a list of NFLs and their names in the background by side-effect

```
createNutritionFactsLabel[name_,
servingSize_,totalCalories_,fatCalories_,
 totalFat_,totalFatPercent_,
 saturatedFat_,saturatedFatPercent_,transFat_,
 cholesterol_,cholesterolPercent_,sodium_,sodiumPercent_,totalCarbohydrates_,totalCarbohydratesPercent_,
 dietaryFiber_,dietaryFiberPercent_,
 sugars_,protein_,proteinPercent_,
 vitaminAPercent_,vitaminCPercent_,calciumPercent_,ironPercent_]:=
(AppendTo[nflNames,name];
nfls[name]={"name"→name,"serving size"→servingSize,"total calories"→totalCalories,"fat calories"→fatCalories,
 "total fat"→totalFat,"% daily total fat"→totalFatPercent,"saturated fat"→saturatedFat,"% daily saturated fat"→saturatedFatPercent,"trans fat"→transFat
```

- Now just build up the NFLs for all the ingredients in the recipe by data-mining again:

■ Olive Oil

```
createNutritionFactsLabel["olive oil", 216 Gram, 1910 Calorie, 1910 Calorie, 216 Gram, 332 Percent, 30 Gram, 149 Percent, 0 Gram,
  0 Gram, 0 Percent, 4 Milli Gram, 0 Percent, 0 Gram, 0 Percent, 0 Gram, 0 Percent, 0 Gram, 0 Percent,
  0 Percent, 0 Percent, 0 Percent, 7 Percent] // gridRules
```

name	olive oil
serving size	Times 216 Gram
total calories	Times 1910 Calorie
fat calories	Times 1910 Calorie
total fat	Times 216 Gram
% daily total fat	Times 332 Percent
saturated fat	Times 30 Gram
% daily saturated fat	Times 149 Percent
trans fat	0
cholesterol	0
% daily cholesterol	0
sodium	Times 4 Gram Milli
% daily sodium	0
total carbohydrates	0
% daily carbohydrates	0
dietary fiber	0
%daily dietary fiber	0
sugars	0
protein	0
% daily protein	0
vitamin A	0
vitamin C	0
calcium	0
iron	Times 7 Percent

▪ Zucchini, summer, with skin, raw

```
createNutritionFactsLabel["zucchini", 124 Gram, 20 Calorie, 2 Calorie,
  0 Gram, 0 Percent, 0 Gram, 0 Percent, 0 Gram,
  0 Gram, 0 Percent, 12 Milli Gram, 1.0 Percent, 4 Gram, 1.0 Percent, 1.0 Gram, 5 Percent,  2 Gram, 2 Gram, 0 Percent,
  5 Percent, 35 Percent,  2 Percent, 2 Percent] // gridRules
```

name	zucchini
serving size	Times 124 Gram
total calories	Times 20 Calorie
fat calories	Times 2 Calorie
total fat	0
% daily total fat	0
saturated fat	0
% daily saturated fat	0
trans fat	0
cholesterol	0
% daily cholesterol	0
sodium	Times 12 Gram Milli
% daily sodium	Times 1. Percent
total carbohydrates	Times 4 Gram
% daily carbohydrates	Times 1. Percent
dietary fiber	Times 1. Gram
%daily dietary fiber	Times 5 Percent
sugars	Times 2 Gram
protein	Times 2 Gram
% daily protein	0
vitamin A	Times 5 Percent
vitamin C	Times 35 Percent
calcium	Times 2 Percent
iron	Times 2 Percent

Table Salt

```
createNutritionFactsLabel["salt", 1. Cup, 0 Calorie, 0 Calorie,
  0 Gram, 0 Percent, 0 Gram, 0 Percent, 0 Gram,
  0 Gram, 0 Percent, 113 174 Milli Gram, 4716 Percent, 0 Gram, 0 Percent, 0 Gram, 0 Percent,  0 Gram, 0 Gram, 0 Percent,
  0 Percent, 0 Percent, 7 Percent, 5 Percent] // gridRules
```

name	salt
serving size	Times | 1. | Cup
total calories	0
fat calories	0
total fat	0
% daily total fat	0
saturated fat	0
% daily saturated fat	0
trans fat	0
cholesterol	0
% daily cholesterol	0
sodium	Times | 113 174 | Gram | Milli
% daily sodium	Times | 4716 | Percent
total carbohydrates	0
% daily carbohydrates	0
dietary fiber	0
%daily dietary fiber	0
sugars	0
protein	0
% daily protein	0
vitamin A	0
vitamin C	0
calcium	Times | 7 | Percent
iron	Times | 5 | Percent

Eggplant, raw

```
createNutritionFactsLabel["eggplant", 82 Gram, 20 Calorie, 1.0 Calorie,
  0 Gram, 0 Percent, 0 Gram, 0 Percent, 0 Gram,
  0 Gram, 0 Percent, 2 Milli Gram, 0 Percent, 5 Gram, 2 Percent, 3 Gram, 11 Percent, 2 Gram, 1.0 Gram, 2 Percent,
  0 Percent, 3 Percent, 1.0 Percent, 1.0 Percent] // gridRules
```

name	eggplant
serving size	Times 82 Gram
total calories	Times 20 Calorie
fat calories	Times 1. Calorie
total fat	0
% daily total fat	0
saturated fat	0
% daily saturated fat	0
trans fat	0
cholesterol	0
% daily cholesterol	0
sodium	Times 2 Gram Milli
% daily sodium	0
total carbohydrates	Times 5 Gram
% daily carbohydrates	Times 2 Percent
dietary fiber	Times 3 Gram
%daily dietary fiber	Times 11 Percent
sugars	Times 2 Gram
protein	Times 1. Gram
% daily protein	Times 2 Percent
vitamin A	0
vitamin C	Times 3 Percent
calcium	Times 1. Percent
iron	Times 1. Percent

▪ Onion, medium, raw

```
createNutritionFactsLabel["onion", 160 Gram, 64 Calorie, 1.0 Calorie,
  0 Gram, 0 Percent, 0 Gram, 0 Percent, 0 Gram,
  0 Gram, 0 Percent, 6 Milli Gram, 0 Percent, 15 Gram, 5 Percent, 3 Gram, 11 Percent, 7 Gram, 2.0 Gram, 0 Percent,
  0 Percent, 20 Percent, 4 Percent, 2 Percent] // gridRules
```

name	onion
serving size	Times 160 Gram
total calories	Times 64 Calorie
fat calories	Times 1. Calorie
total fat	0
% daily total fat	0
saturated fat	0
% daily saturated fat	0
trans fat	0
cholesterol	0
% daily cholesterol	0
sodium	Times 6 Gram Milli
% daily sodium	0
total carbohydrates	Times 15 Gram
% daily carbohydrates	Times 5 Percent
dietary fiber	Times 3 Gram
%daily dietary fiber	Times 11 Percent
sugars	Times 7 Gram
protein	Times 2. Gram
% daily protein	0
vitamin A	0
vitamin C	Times 20 Percent
calcium	Times 4 Percent
iron	Times 2 Percent

Bell Pepper, sweet, yellow, raw

```
createNutritionFactsLabel["bell pepper", 186 Gram, 50 Calorie, 3.0 Calorie,
  0 Gram, 1.0 Percent, 0 Gram, 0 Percent, 0 Gram,
  0 Gram, 0 Percent, 4 Milli Gram, 0 Percent, 12 Gram, 4 Percent, 2 Gram, 7 Percent, 2 Gram, 2 Gram, 0 Percent,
  7 Percent, 569 Percent, 2 Percent, 5 Percent] // gridRules
```

name	bell pepper
serving size	Times 186 Gram
total calories	Times 50 Calorie
fat calories	Times 3. Calorie
total fat	0
% daily total fat	Times 1. Percent
saturated fat	0
% daily saturated fat	0
trans fat	0
cholesterol	0
% daily cholesterol	0
sodium	Times 4 Gram Milli
% daily sodium	0
total carbohydrates	Times 12 Gram
% daily carbohydrates	Times 4 Percent
dietary fiber	Times 2 Gram
%daily dietary fiber	Times 7 Percent
sugars	Times 2 Gram
protein	Times 2 Gram
% daily protein	0
vitamin A	Times 7 Percent
vitamin C	Times 569 Percent
calcium	Times 2 Percent
iron	Times 5 Percent

Stewed Tomato

```
createNutritionFactsLabel["stewed tomato", 101 Gram, 80 Calorie, 24.0 Calorie,
  3 Gram, 4 Percent, 1.0 Gram, 3 Percent, 0 Gram,
  0 Gram, 0 Percent, 460 Milli Gram, 19 Percent, 13 Gram, 4 Percent, 2 Gram, 7 Percent, 0 Gram, 2 Gram, 0 Percent,
  13 Percent, 31 Percent, 3 Percent, 6 Percent] // gridRules
```

name	stewed tomato
serving size	Times 101 Gram
total calories	Times 80 Calorie
fat calories	Times 24. Calorie
total fat	Times 3 Gram
% daily total fat	Times 4 Percent
saturated fat	Times 1. Gram
% daily saturated fat	Times 3 Percent
trans fat	0
cholesterol	0
% daily cholesterol	0
sodium	Times 460 Gram Milli
% daily sodium	Times 19 Percent
total carbohydrates	Times 13 Gram
% daily carbohydrates	Times 4 Percent
dietary fiber	Times 2 Gram
%daily dietary fiber	Times 7 Percent
sugars	0
protein	Times 2 Gram
% daily protein	0
vitamin A	Times 13 Percent
vitamin C	Times 31 Percent
calcium	Times 3 Percent
iron	Times 6 Percent

▪ Black Pepper (spices, pepper, black)

```
createNutritionFactsLabel["black pepper", 1. Tablespoon, 16 Calorie, 2 Calorie, 0 Gram, 0 Percent, 0 Gram, 0 Percent, 0 Gram,
  0 Gram, 0 Percent, 3 Milli Gram, 0 Percent, 4 Gram, 1. Percent, 2 Gram, 7 Percent,  0 Gram, 1. Gram, 0 Percent,
  0 Percent, 2 Percent, 3 Percent, 10 Percent] // gridRules
```

name	black pepper
serving size	Times | 1. | Tablespoon
total calories	Times | 16 | Calorie
fat calories	Times | 2 | Calorie
total fat	0
% daily total fat	0
saturated fat	0
% daily saturated fat	0
trans fat	0
cholesterol	0
% daily cholesterol	0
sodium	Times | 3 | Gram | Milli
% daily sodium	0
total carbohydrates	Times | 4 | Gram
% daily carbohydrates	Times | 1. | Percent
dietary fiber	Times | 2 | Gram
%daily dietary fiber	Times | 7 | Percent
sugars	0
protein	Times | 1. | Gram
% daily protein	0
vitamin A	0
vitamin C	Times | 2 | Percent
calcium	Times | 3 | Percent
iron	Times | 10 | Percent

▪ Dried Basil (spices, basil, dried)

```
createNutritionFactsLabel["dried basil", 1. Teaspoon, 1.0 Calorie, 0 Calorie, 0 Gram, 0 Percent, 0 Gram, 0 Percent, 0 Gram,
  0 Gram, 0 Percent, 0 Gram, 0 Percent, 0 Gram, 0 Percent,
  0 Gram, 1.0 Percent, 0 Gram, 0 Gram, 0 Percent,
  1.0 Percent, 1.0 Percent, 1.0 Percent, 1.0 Percent] // gridRules
```

name	dried basil
serving size	Times | 1. | Teaspoon
total calories	Times | 1. | Calorie
fat calories	0
total fat	0
% daily total fat	0
saturated fat	0
% daily saturated fat	0
trans fat	0
cholesterol	0
% daily cholesterol	0
sodium	0
% daily sodium	0
total carbohydrates	0
% daily carbohydrates	0
dietary fiber	0
%daily dietary fiber	Times | 1. | Percent
sugars	0
protein	0
% daily protein	0
vitamin A	Times | 1. | Percent
vitamin C	Times | 1. | Percent
calcium	Times | 1. | Percent
iron	Times | 1. | Percent

▪ Sugar (sugars, granulated [sucrose])

```
createNutritionFactsLabel["sugar", 2 Gram, 11 Calorie, 0 Calorie,
  0 Gram, 0 Percent, 0 Gram, 0 Percent, 0 Gram,
  0 Gram, 0 Percent, 0 Gram, 0 Percent,
  3 Gram, 1.00 Percent, 0 Gram, 0 Percent, 3 Gram,
  0 Gram, 0 Percent,
  0 Percent, 0 Percent, 7 Percent, 5 Percent] // gridRules
```

name	sugar
serving size	Times 2 Gram
total calories	Times 11 Calorie
fat calories	0
total fat	0
% daily total fat	0
saturated fat	0
% daily saturated fat	0
trans fat	0
cholesterol	0
% daily cholesterol	0
sodium	0
% daily sodium	0
total carbohydrates	Times 3 Gram
% daily carbohydrates	Times 1. Percent
dietary fiber	0
%daily dietary fiber	0
sugars	Times 3 Gram
protein	0
% daily protein	0
vitamin A	0
vitamin C	0
calcium	Times 7 Percent
iron	Times 5 Percent

- **Pasta, fresh-refrigerated, plain, as purchased**

```
createNutritionFactsLabel["pasta",
  128 Gram, 369 Calorie, 25 Calorie,
  3 Gram, 5 Percent, 0 Gram, 2 Percent, 0 Gram,
  93 Gram, 31 Percent, 33 Milli Gram, 1.0 Percent,
  70 Gram, 23 Percent, 0 Gram, 0 Percent, 0 Gram,
  14 Gram, 0 Percent,
  1.0 Percent, 0 Percent, 2 Percent, 24 Percent] // gridRules
```

name	pasta
serving size	Times 128 Gram
total calories	Times 369 Calorie
fat calories	Times 25 Calorie
total fat	Times 3 Gram
% daily total fat	Times 5 Percent
saturated fat	0
% daily saturated fat	Times 2 Percent
trans fat	0
cholesterol	Times 93 Gram
% daily cholesterol	Times 31 Percent
sodium	Times 33 Gram Milli
% daily sodium	Times 1. Percent
total carbohydrates	Times 70 Gram
% daily carbohydrates	Times 23 Percent
dietary fiber	0
%daily dietary fiber	0
sugars	0
protein	Times 14 Gram
% daily protein	0
vitamin A	Times 1. Percent
vitamin C	0
calcium	Times 2 Percent
iron	Times 24 Percent

▪ Parmesan Cheese (Cheese, parmesan, grated)

```
createNutritionFactsLabel["parmesan cheese", 100 Gram, 431 Calorie, 251 Calorie, 29 Gram, 44 Percent, 17 Gram, 86 Percent, 0 Gram,
  88 Gram, 29 Percent, 1529 Milli Gram, 64 Percent,
  4 Gram, 1.00 Percent, 0 Gram, 0 Percent, 1 Gram,
  38 Gram, 0 Percent,
  9 Percent, 0 Percent, 111 Percent, 5 Percent] // gridRules
```

name	parmesan cheese
serving size	Times 100 Gram
total calories	Times 431 Calorie
fat calories	Times 251 Calorie
total fat	Times 29 Gram
% daily total fat	Times 44 Percent
saturated fat	Times 17 Gram
% daily saturated fat	Times 86 Percent
trans fat	0
cholesterol	Times 88 Gram
% daily cholesterol	Times 29 Percent
sodium	Times 1529 Gram Milli
% daily sodium	Times 64 Percent
total carbohydrates	Times 4 Gram
% daily carbohydrates	Times 1. Percent
dietary fiber	0
%daily dietary fiber	0
sugars	Gram
protein	Times 38 Gram
% daily protein	0
vitamin A	Times 9 Percent
vitamin C	0
calcium	Times 111 Percent
iron	Times 5 Percent

4. & 5. HOW TO ADD UNIT-NFLS

Canonicalize Units

- Convert anything compatible with Gram to Gram
- Convert rules about volumes to rules about weights
  - Another rule that rewrites rules

```
canonicalizeUnits[nfl_] :=
 Map[Function[rule, rule[[1]] → Quiet[N@Convert[rule[[2]], Gram]]],
  (nfl /. {
     (* pattern to match against victim rule *)
     (keyWithVolume_ → (* green arrow is part of victim rule *)
        amount_?NumberQ * volume : (Teaspoon | Tablespoon | Cup | FluidOunce | Pint | Gallon)) :→ (* pink arrow is part of the rewriting meta-rule *)
      (* this is the resulting new rule with green arrow again *)
      keyWithVolume → amount * volume *
        gramPerTargetVolumeFromDensity[volume, density["name" /. nfl]]})]
```

NORMALIZE, SCALE, ADD

```
nflList = Map[Function[name, nfls[name]], nflNames];
```

```
canonicalizedNfls = canonicalizeUnits /@ nflList;
```

```
norms = ("serving size" / Gram /. # &) /@ canonicalizedNfls
```

```
{216., 124., 273.12, 82., 160., 186., 101., 6.3, 1., 2., 128., 100.}
```

This is vector scale!

```
scaleNfl[nfl_, scalar_] :=
 Map[Function[line, If[line[[1]] === "name",
     line, (* skip the name line *)
     line[[1]] → line[[2]] * scalar]], nfl]
```

```
normalizedNfls = MapThread[scaleNfl, {canonicalizedNfls, 1 / norms}];
```

```
scaledNfls = MapThread[scaleNfl, {normalizedNfls, recipeInGrams / Gram / nflNames}];
```

This is vector sum!

```
sumNfls[nfl1_, nfl2_] :=
 MapThread[Function[{line1, line2},
   If[line1[[1]] === line2[[1]], (* don't add up dimensions that don't match *)
    line1[[1]] → (line1[[2]] + line2[[2]]) // Chop,
    Throw["foo"]]],  {nfl1, nfl2}]
```

FINAL RECIPE, FEEDING SIX

```
scaleNfl[Fold[sumNfls, First[scaledNfls], Rest[scaledNfls]], 1 / 6] // gridRules
```

name	Plus	bell pepper	black pepper	dried basil	eggplant	olive oil	onion	parmesan cheese	pasta	salt	stewed tomato	sugar	zucchini
serving size	Times 368.02 Gram												
total calories	Times 309.72 Calorie												
fat calories	Times 58.413 Calorie												
total fat	Times 6.546 Gram												
% daily total fat	Times 9.8998 Percent												
saturated fat	Times 1.5959 Gram												
% daily saturated fat	Times 7.5358 Percent												
trans fat	0												
cholesterol	Times 44.422 Gram												
% daily cholesterol	Times 14.795 Percent												
sodium	Times 1.7696 Gram												
% daily sodium	Times 73.6 Percent												
total carbohydrates	Times 53.543 Gram												
% daily carbohydrates	Times 17.71 Percent												
dietary fiber	Times 6.8463 Gram												
%daily dietary fiber	Times 25.73 Percent												
sugars	Times 5.8525 Gram												
protein	Times 12.1 Gram												
% daily protein	Times 2.7658 Percent												
vitamin A	Times 13.434 Percent												
vitamin C	Times 107.62 Percent												
calcium	Times 11.903 Percent												
iron	Times 19.675 Percent												