

SCHEMA for Dummies

How to write a workout in one line - and gradually learn to
understand your own load

Serega Photoskiff - creator of the Schema language and SchemaFit Timer

2026

Contents

About this book	2
Part I - The first five minutes	3
1. Start with a workout	3
2. What if there is no recovery?	3
3. Repeat it	3
4. Remove the final recovery	3
Part II - Load, repetitions and tempo	4
5. Add load	4
6. The most important rule after @	4
7. Intensity is non-linear	4
8. Prescribe repetitions with r	5
9. Prescribe tempo with t	5
10. Read a dense-looking atom	5
Part III - Assemble a workout	6
11. The atom	6
12. Several atoms in sequence	6
13. Groups	6
14. Labels with !1, !2,	6
15. Give a block a time budget	6
Part IV - Leave room for reality	7
16. An unknown number of rounds	7
17. Recorder mode	7
18. Materialisation	7

Part V - Why an incomplete Schema still works	8
19. The stencil principle	8
20. Why /0 matters	8
21. Why # matters	8
22. Missing intensity is not zero	9
23. Calibrate before fixing intensity	9
Part VI - Build a personal map of load	9
24. More than a workout catalogue	9
25. A verified Schema is a small piece of knowledge	10
26. Learn from every pass	10
Part VII - Controlled variability	10
27. Progression	10
28. Tempo ramps	10
29. Map the ramp to intensity	11
30. Controlled randomness	11
31. Adaptive work	11
Part VIII - Read and write with confidence	12
32. Constants	12
33. Arithmetic	12
34. Read a frightening Schema in layers	12
35. Build a Schema one thought at a time	12
36. Six common mistakes	13
One-page reference	13
Epilogue - At first, you collect Schemas	14

About this book

This is not a language specification and it is not a syntax exam. The book is arranged so that every few minutes you gain one new tool that you can already use in a workout.

First there is one simple line. Then one more symbol. Then one more possibility. Technical terms appear only when they genuinely make things easier.

The central idea is simple: **do not try to learn all of Schema at once**. Ask, “What else do I want to tell the timer?” Then add only that.

30/30#

Part I - The first five minutes

1. Start with a workout

Suppose the workout is:

work for 30 seconds, rest for 30 seconds.

In Schema:

30/30#

The first number is work. The second is recovery.

WORK / RECOVERY #

Therefore:

60/20#

means: work for 60 seconds, recover for 20 seconds, then finish. The # symbol says that the Schema ends here.

2. What if there is no recovery?

Write the zero explicitly:

60/0#

This means 60 seconds of work, zero seconds of recovery, then the end.

This is not pointless formality. When something is omitted, the timer or another execution environment may fill it from its current settings. /0 says “there is definitely no recovery”, while # says “finish exactly here”.

3. Repeat it

To perform 30 seconds of work and 30 seconds of recovery five times:

30/30*5#

The * symbol means “repeat”, so *5 means five rounds.

4. Remove the final recovery

With:

40/30*5#

the timer performs 30 seconds of recovery after the fifth work interval as well. If the session ends immediately, that recovery may be unnecessary:

40/30*5`#

The backtick removes only the final recovery from the final round.

Part II - Load, repetitions and tempo

5. Add load

Thirty seconds of walking and 30 seconds of maximal sprinting are not the same load. Schema uses @ to describe the parameters of an atom:

30/30@8#

This is 30 seconds of work at intensity 8, followed by 30 seconds of recovery.

Schema uses an intensity scale from 0 to 10. A value of 10 is approximately 100% of the effort currently available: an absolute maximum. If you do not yet know the right number, that is normal. One purpose of working with Schema is to discover your intensities from repeatable sessions rather than invent them.

6. The most important rule after @

15/15@8/2#

Before @, 15/15 means work / recovery. After @, the same correspondence applies: 8/2 means work parameters / recovery parameters.

- Work: 15 seconds, intensity 8.
- Recovery: 15 seconds, intensity 2.

If there is no second side after @, the parameters apply to work only:

30/30@8#

7. Intensity is non-linear

It is tempting to treat the 0-10 scale as a ruler with equal steps. The body does not behave that way. Intensities 1-3 may describe easy or moderate work that can continue for a long time. A value around 9 is already severe and sustainable only for a limited period. Near 10, the cost of each additional step rises sharply.

Approximate examples might be:

- 9 - a matter of minutes;

- 9.9 - a matter of tens of seconds;
- 10 - an absolute maximum, perhaps around 20 seconds.

These are not medical limits or universal time values. They depend on the person, movement and conditions. The useful principle is simply that **the upper end is strongly non-linear**.

Decimal values use a full stop:

30/30@8.45#

30.5/10#

8. Prescribe repetitions with r

If the goal is ten repetitions within a 30-second work interval:

30/30@r10#

r stands for repetitions. Intensity and a repetition target can be combined:

30/30@8r10#

9. Prescribe tempo with t

Sometimes the important instruction is not “perform 20 repetitions in total” but “move at 20 repetitions per minute”. Schema uses **t** for tempo:

60/0@t20#

This is 60 seconds of work at a tempo of 20 repetitions or movements per minute.

Do not confuse r and t:

r20 = 20 repetitions in total

t20 = a rate of 20 per minute

On the same side of an atom, r and t are mutually exclusive. Intensity is a separate layer and may accompany either:

60/0@8r20#

60/0@8t20#

10. Read a dense-looking atom

15/15@8.5t40/2t20#

- Work: 15 seconds, intensity 8.5, tempo 40 per minute.
- Recovery: 15 seconds, intensity 2, tempo 20 per minute.

The same work/recovery correspondence operates on both sides of @.

Part III - Assemble a workout

11. The atom

An **atom** is one executable piece of a workout:

```
30/30
60/0@t20
15/15@8.5t40/2t20
```

12. Several atoms in sequence

```
30/30 20/40#
```

The timer performs 30/30 first, then 20/40. Schema is read from left to right.

13. Groups

To repeat a whole sequence, put it in parentheses:

```
(30/30 20/40)*3#
```

Parameters such as intensity, **r** and **t** belong to individual atoms, not to the entire group:

```
(30/30@8 20/40@8)*3#
```

14. Labels with !1, !2, ...

Labels tell the timer what happens during a segment:

```
15!1/0 15!2/0# 1=pull-ups 2=push-ups
```

!1 means “use label 1”; !2 means “use label 2”. The label text lives in the metadata after #. A label names a segment but does not change its duration, tempo or intensity.

15. Give a block a time budget

dur: sets a local duration:

```
dur:600(40/20*100)#
```

Execute the 40/20 cycle, but stop the whole block at 600 seconds. The high repeat count simply ensures that the block cannot run out of planned cycles first.

Likewise, **reps:** sets a repetition budget:

```
reps:100(30/0@r10*20)#
```

Continue until the prescribed total volume reaches 100 repetitions.

```
dur:N(...) = time budget
reps:N(...) = repetition budget
```

Part IV - Leave room for reality

16. An unknown number of rounds

If each round is known but today's number of rounds is not:

```
180/0*~#title=Open E3MOM
```

~ means “this particular value will be determined later”. Here everything is known except the number of rounds.

You may still prescribe tempo or repetitions:

```
180/0@t20*~#
180/0@r50*~#
```

17. Recorder mode

```
~/0#
```

Even the work duration is unknown in advance. Start the timer and do what the session requires. The timer imposes no interval structure; it records the session and any available sensor data.

18. Materialisation

Suppose you execute:

```
180/0*~#
```

and complete seven rounds. The unknown value has now become concrete:

```
180/0*7#
```

This is **materialisation**: values that were deferred become known. The result can be saved as a new active workout.

Four useful terms:

- **Active** - the structure is known in advance: 30/30*5#.

- **Deferred** - something will be decided during execution: 30/30*~#.
- **Materialised** - deferred values have become concrete.
- **Realised** - what the athlete actually performed.

The plan and reality are not always identical, and that difference is useful data.

Part V - Why an incomplete Schema still works

19. The stencil principle

SchemaFit Timer has ordinary controls for work, recovery, rounds, total duration and work/recovery intensities. Together these values form a complete underlying Schema. Think of it as a solid backing layer.

The user's text Schema sits above it like a stencil. Values written explicitly cover the corresponding values below. Missing values are holes through which the backing layer remains visible.

The backing layer is always complete; the user Schema may be partial.

Suppose the controls currently define:

```
work 30 | recovery 20 | rounds 5 | work intensity 7 | recovery intensity 2
```

If the user writes only a new work duration, only that field is replaced. Everything else may still come from the backing layer.

20. Why /0 matters

An omitted recovery may be supplied by the backing layer. Explicit zero closes that hole:

```
20/0
```

Recovery really is zero.

21. Why # matters

```
20/0#
```

The workout ends here. Without an explicit end, an execution environment may be allowed to continue using its own settings.

22. Missing intensity is not zero

30/30*10#

No intensity is written, but that does not mean intensity 0. It is simply missing and may be supplied by the backing layer so that the timer can construct a load and heart-rate model.

23. Calibrate before fixing intensity

Suppose tempo is known but its true intensity for you is not:

60/0@t20*10#

Do not invent a number. Let the underlying settings supply an initial estimate, perform the session, inspect the measured heart rate, and tune the model. When repeated sessions support a value, write it into the Schema:

60/0@7.4t20*10#

The value is now part of the portable workout rather than a local default.

Part VI - Build a personal map of load

24. More than a workout catalogue

Over time, useful Schemas accumulate relationships between load and concrete conditions:

- tempo;
- target repetitions;
- implement weight;
- exercise;
- movement speed;
- duration;
- the athlete's current state.

Not every condition has to be encoded by one symbol. The important point is that a verified Schema, stored with its context, becomes a measured point on a personal map of load.

25. A verified Schema is a small piece of knowledge

60/007.4t20*10#

After several checks, this line says more than “work for ten minutes at tempo 20”. It says:

Under these conditions, this work normally corresponds to approximately this load and this physiological response for me.

If the model has repeatedly matched measured heart rate, the Schema becomes a calibrated reference point.

26. Learn from every pass

Once several tempos have been verified for one movement and weight - perhaps t18, t20 and t22 - a new t21 or a ramp t[18->22] no longer begins from guesswork.

design -> perform -> compare -> refine -> save -> reuse

Ask not only “What workout do I want to do?” but also:

What will this workout teach me about my load?

Part VII - Controlled variability

27. Progression

Instead of writing:

30/30
25/35
20/40

write:

30/30->[for:3,delta:-5/+5]#

delta describes the change: work -5 seconds, recovery +5 seconds. for:3 produces three values. A progression may also continue to: a target value.

28. Tempo ramps

A tempo ramp changes tempo inside one atom:

60/10@t[15->25]#

During the 60-second work period, tempo rises gradually from 15 to 25 per minute. Reverse the values to decelerate:

```
60/10@t[25->15]#
```

A step may be supplied with **d:** or **delta::**

```
60/10@t[15->25 d:2]#
```

29. Map the ramp to intensity

```
60/10@t[15->25 i:6->9]#
```

The tempo changes from 15 to 25. The **i:6->9** corridor says that, for this athlete in these conditions, the two ends correspond approximately to intensities 6 and 9.

This does **not** claim that tempo 25 always means intensity 9. Change the exercise, weight, duration or athlete and the mapping changes. Schema records a relationship in a specific context.

Intermediate intensity values are derived with the non-linear nature of the scale in mind. They need not increase by equal numerical steps.

A complete example:

```
58/20@t[18->26 i:4->8]*10`#title=accelerating high swing
```

30. Controlled randomness

```
random(
  30/00@9t220
  30/00@6t140
  30/00@7t180
  turb:7 dur:580 min:8
)#
```

You define the permitted atoms and constraints; the runtime materialises one variable sequence within them. **random** is not chaos. It is controlled variability.

Add a **seed** when the same generated sequence must be repeatable. Think of the seed as the identifier of one particular random course.

31. Adaptive work

```
300/00@t180^#
```

^ marks an adaptive atom. The original plan is tempo 180, but realised tempo may be changed during execution. The timer keeps the intended and realised forms separate; the plan is not rewritten after the fact.

Part VIII - Read and write with confidence

32. Constants

Give a repeated value a name:

```
[N=5]30/30*N#
```

N is 5, so *N is equivalent to *5. Constants are particularly useful when one parameter controls several places.

33. Arithmetic

Numeric fields may use ordinary arithmetic and parentheses:

```
[N=5]30/(N*6)#
```

N*6 is 30, so the realised atom is 30/30. Values that are conceptually integral, such as a repetition target, should remain integral; the timer should not silently decide how to round `r10.37`.

34. Read a frightening Schema in layers

```
58/2@t[18->26 i:4->8]*10`#
```

- 58/2 - work and recovery;
- @ - parameters begin;
- t[18->26] - tempo ramp;
- i:4->8 - intensity correspondence at the ramp ends;
- *10 - ten rounds;
- backtick - remove final recovery;
- # - finish.

One intimidating line has become seven familiar ideas.

35. Build a Schema one thought at a time

```
30/30#
30/30*5#
30/30@8*5#
30/30@8r10*5#
30/30@8r10*5`#
```

Each line adds one new instruction. The useful question is not “How do I

memorise Schema?” but “What else do I want to tell the timer?”

36. Six common mistakes

- Mixing the sides: 15/15@8/2# means work 8 / recovery 2, not a ramp from 8 to 2.
- Mixing r and t: r10 is ten repetitions; t10 is ten per minute.
- Treating absence as zero: 30/30# does not prescribe intensity 0.
- Using a decimal comma inside Schema: write @8.5, not @8,5.
- Treating ~ as total uncertainty: in 180/0@t20*~#, only the number of rounds is unknown.
- Trying to understand a long Schema all at once: read it in layers.

One-page reference

Schema	Meaning
30/30#	30 seconds work / 30 recovery
300/0#	continuous work, no recovery
30/30*5#	five rounds
30/30*5`#	five rounds, no final recovery
30/30@8#	work intensity 8
30/30@8/2#	work intensity 8 / recovery intensity 2
30/30@r10#	target 10 repetitions
30/30@t20#	tempo 20 per minute
30/30@8r10#	intensity 8 and target 10 repetitions
(30/30 20/40)*3#	repeat a two-atom group three times
dur:600(40/20*100)#	give the block a 600-second budget
reps:100(30/0@r10*20)#	give the block a 100-repetition budget
180/0*~#	as many rounds as chosen during execution
~/0#	Recorder mode
30/30->[for:3,delta:-5/+5]#	progression: 30/30, 25/35, 20/40
60/10@t[15->25]#	ramp tempo from 15 to 25
300/0@t180^#	adaptive active atom

Epilogue - At first, you collect Schemas

The book began with one line:

30/30#

Then came load, tempo, repetitions, groups, deferred values, ramps and variability. But the more interesting stage begins when Schemas stop being merely workout records.

One verified Schema says, “this is something I already know about myself”. Another adds one more point:

- this tempo with this weight is comfortable for me;
- here the load becomes severe;
- this is how my heart rate normally responds;
- this used to be near my limit and is now ordinary work;
- here I have only begun to investigate.

Gradually, a workout catalogue becomes a catalogue of verified models of your own body.

At first, you collect Schemas. Then you realise that you are actually collecting knowledge about yourself.

Schema does not tell an athlete which workout they must do. It lets them record the workout they chose, reproduce it, test it on themselves and learn something new from it.

In formal terms, **Schema is a language of controlled variability in training load, and SchemaFit Timer is an environment for its repeatable execution and physiological validation.**

But it still begins with one line:

30/30#