Let *n* be the number of possible attacking creatures, and *d* the total number of defenders.

First and foremost, determine the *global maximum*. This is either the maximum set by static effects, or the total number of creatures controlled by the active player, whichever is smaller. If the number of creatures that may attack a specific defender is constrained for each available defender, this number is set to the sum of all those constraints. O(*d*)
If this number is 0, creatures can't attack this turn.

Then construct the restrictions ("can't attack [unless]" and the like) for each creature controlled by the active player, and do the same for the requirements ("attacks [player] each {turn|combat} if able"). For the restrictions, just consider whether a creature has a particular restriction. For the requirements, count the number of requirements the creature has to attack (in general as well as to specific defenders), and the number of requirements the creature puts on other creatures to attack (in general as well as for specific other creatures). Remove any requirements on attacking a defender that creature can't attack. O(*dn*)

Next, filter out creatures that can't attack this combat. This includes creatures with defender, creatures with conflicting restrictions (ie. "can't attack alone" and "can only attack alone"), and creatures that can't attack a set of specific defenders which happens to including all possible defenders. O(*n*)

Then filter creatures that require something that can't ever be true (such as requiring more creatures to attack than the global maximum, or requiring a creature with a certain property to attack while there are no more left that are capable of attacking). O(*n*^2)
If at this point no creatures remain, creatures can't attack this turn. Otherwise, redetermine the global maximum by taking into account the current number of valid attackers. O(1)

Now come the more specific restrictions.

- ★ For each creature that can only attack alone (or *all* available creatures if the global maximum is 1), ultimately consider that creature attacking a defender for which is has the most requirements to attack that player. O(*n*)

Remove all these creatures from consideration, as they have no other legal means of attacking.

-- (see end of document)

Next are the *predicate restrictions*. These include requirements that one or more other creatures attack which together fulfill a certain predicate, for example one of them being black or green, or there being at least two total. For each creature *c* with one or more predicate restrictions, do the following:

- Construct a chain. A chain is constructed by requiring a number of other creatures, that together fulfill each of *c*'s predicate restrictions, to attack. When choosing creatures, proceed like this:
    - Determine each *minimal* set of creatures that fulfill all of *c*'s predicate restrictions. A set is *minimal* if removing a creature would violate one of *c*'s predicate restrictions.
        - Loop over all possible attackers, and select each that fulfills at least one predicate restriction of *c*.
        -
    - Repeat this procedure for each creature in the chain having a predicate restriction that's not yet fulfilled by the current chain.
- Fill each chain using this procedure, until all chains have all their creatures fulfill all their predicate restrictions.
- For any chain whose size exceeds the global maximum, discard it.
- If the minimal way of doing this for a certain creature exceeds the global maximum, remove this creature from consideration altogether and restart the chaining procedure.

Having constructed one or more chains, for each attacker with a predicate restrictions, consider the subset of chains containing that attacker which fulfill the most requirements. All these together (for all possible attackers) is the set of **base chains**.
Remove each creature with a predicate restriction from further consideration (except when considering the base chains). O(*n*)

Now construct the final sets of attackers, consisting of:

- Consider each base chain, appended with more attackers as follows:
    - Consider the case where each remaining possible attacker attacks the defender it has the most requirements to attack (including requirements because creatures already in the chain require the creature to attack). Do this in order of decreasing total number of requirements. If two or more defenders are tied, choose one without restrictions if possible. In case the most favourable defender of the creature has already reached its maximum number of attackers, see "Defender Handling", below. O(*dn*)
    - If at any point this reaches the global maximum, just stop at that point.
- Do the same for the empty base chain. O(*dn*)
- Consider creatures attacking alone, as determined under ★. O(*n*)

Take whichever of these fulfills the most requirements.

Defender Handling

In case an attacking creature is to be added to a set of attackers and the most favourable defender of that creature has already reached its maximum number of attackers, let *x* be the difference between the requirements on attacking that defender and the next-to-most favourable (or 0 if there is none).

Calculate *y* for each other creature *c* in the current set of attackers:

- Let *y* be the smallest difference between the requirements on *c* attacking its current defender and it attacking any other defender.
- If that defender has already reached its maximum, try defender handling again, using *c* as the creature that's to be added.

If *x* is greater than *y* for any other creature in the current set of attackers, swap the defenders of those creatures. Otherwise, repeat the procedure but remove that the most favourable defender from consideration. If there are no more favourable defenders, treat it as having no requirements.

This method should ensure that the creature

NEW APPROACH -- REPLACES ALL STARTING AT “now come the predicate restrictions”

Sort all creatures by their total number of requirements (per defender). This is a list of triplets: (Card attacker, GameEntity defender, int constraints). Traverse this list from most to fewest requirements, taking the following into account:

- After adding the first creature, add a requirement of 1 (to each defender) to each creature with ”CARDNAME attacks if another creature attacks”.
- If a creature is encountered which causes other creatures to attack:
    - If the global maximum is less than the number of possible attackers, branch: check both the case where we include the creature and add a requirement to each creature on its “causes to attack list” as well as the case where we skip it.
    - Otherwise, check only the case where we include the creature.
- If a creature with a predicate restriction is encountered and that restriction is not yet fulfilled by either itself or one of the reserved cards:
    - Reserve the next creature in line that does fulfil the predicate. Add it to the reserved list and decrease the max number of creatures we can still add by one.
    - If the global maximum is less than the number of possible attackers, branch: check both the case where we include the predicate creature and the case where we skip it. Otherwise, check only the case where we include the creature.
- Remember to remove a creature’s requirements completely once it’s added (be it the final list or the reserved list) and start again at the top of the list, as the list may have changed.
- If the only way for a branch to continue is to add too many creatures to the list + reserved list total, discard it.
- If at the end any creature requires “another creature” or “at least two other creatures” to attack, arbitrarily add sufficient creatures to fulfil the constraint (which is possible; see preliminary filters).